UNIVERSITY COLLEGE LONDON

Traffic Engineering for Multiservice IP Networks

Submitted to the Department of Electronic & Electrical Engineering in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Electronic & Electrical Engineering

Jonas Griem

Department of Electronic & Electrical Engineering
University College London
London, England 2007

Supervisor: **Professor Chris Todd**

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Traffic Engineering for Multiservice IP Networks

Department of Electronic & Electrical Engineering
University College London
WC1E 7JE London, England
Phone: +44 (0)20–7679 7610
Web: http://www.ee.ucl.ac.uk/~jgriem

Email: jonas@griem.de

Acknowledgements

There is a whole spectrum of people I would like to thank for helping me along the way to accomplishing my goal of attaining a Ph.D.

First, I would like to thank my supervisors Chris Todd and Miguel Rio for their continuing assistance and encouragement as well as their good work in securing the research funding that supported me along my way. I would also like to thank my examiners Ian Henning and Javier Barria for helping me make this a stronger thesis.

A special thanks goes to David Griffin for being a great friend and mentor. This thesis would not have been possible without his vast knowledge of the field and his piercing attention to detail. A big thank you goes to Jason Spencer, who taught me much of what I know programming and scripting. Thank you also to Aleksandar Lazarevic, whose sharp and clarifying insights were always available to me. Finally, I thank Bosse Myhr for frequently and successfully scaring my into working harder.

My gratitude also goes to my colleagues, who continuously supported me during my time here. In particular I would like to thank Nikos Vardalachos, Alvin Tan, Raul Landa, Hamed Hadaddi, Venus Shum, Ibiso Wokoma, Lawrence Latif, Spiros Spirou and Eleni Mykoniati. You have made my time here a fun and pleasant experience.

None of the success would have been possible without the support of my family. Uwe and Heleni, my parents were unconditionally supportive and helped me along the way as much as they could. Their belief in my success kept me on course over this four and a half year long journey that has now come to a close.

Jonas Griem
9th of November 2007

Traffic Engineering for Multiservice IP Networks

Jonas Griem
Department of Electrical & Electronic Engineering, University College London

Abstract

The central cause of strain on today's IP network infrastructure stems from the industries striving to extend the network to carry voice and other advanced services that it was not designed for. This thesis develops the reasons why more sophisticated mechanisms are required for ensuring that the best-effort infrastructure becomes both predictable and controllable, if the current IP infrastructure is to replace circuit switched networks.

The main theme of the thesis are the IP traffic engineering (IPTE) traffic engineering algorithms that were developed by the author. The goal of the proposed approach is to compute a set of Open Shortest Path First (OSPF) link weights to balance network load, while honouring bandwidth, hop count and propagation delay constraints of the traffic. The proposed solution is built on classical OSPF routing and additionally uses the multi topology concepts that were recently proposed as an extension to OSPF. Using Multi Topology OSPF (mt-OSPF), multiple routing planes can be implemented. Each Multi Topology (MT) routing plane is assigned its own link metrics and thus traffic can be routed independently. Given a traffic demand matrix and the network topology, the algorithm computes a set of link weights using a search heuristic. The optimisation is cost function based, so that individual constraints can be taken into account per routing plane. Using this approach, constraint information remains in the "offline" IPTE algorithms and thus no extra constraint awareness is required at layer 3. Since recent Cisco implementations of mt-OSPF and Multi Topology IS-IS (M-ISIS) provide the required multi topology routing support, no major changes at the router level are required for the approach to be realised.

Extensive simulations presented in this thesis show that IPTE has the potential to provide the differentiated routing and load balancing that it was designed for. In addition the simulations show that load is balanced more evenly across the network than with standard shortest path routing on inverse capacity link weights.

A network management system is discussed that acts as the binding element for all enabling components. A simplified architecture for such a management system is presented and it is discussed how several traffic

engineering mechanisms could coexist with IPTE on one network infrastructure. The use of the authors IPTE approach in the context of this management system is discussed in depth: Dangers associated with potential network disruption caused by frequent link weight modifications are analysed from both intra- and inter-domain point of view. A strategy for solving the problem of disruptions caused by link weight changes is presented by using several network planes to migrate traffic, rather than causing disruptions to a "live" plane through link weight modification and resulting OSPF routing table updates. The strategy is simulated, showing that any detrimental effects of the transition are avoidable. However, the work is ongoing and the results presented are indicative.

Overall the thesis presents an approach for traffic engineering using traditional routing techniques, to (1) preserve the original intrinsic advantages of the IP design and (2) help ready IP networking for the much more restricting requirements of the future mutiservice network. It presents algorithms, simulations and frameworks showing how the work fits into the current IP networking world.

Contents

List of Figures

List of Tables

Glossary

AGAVE A liGhtweight Approach for Viable End-to-end IP-based QoS Services

API Application Programming Interface

ARPA United States Defence Advanced Research Project Agency

ASON Automatic Switched Optical Network

ATM Asynchronous Transfer Mode

BGP Border Gateway Protocol

CCITT Comit Consultatif International Tlphonique et Tlgraphique

CORBA Common Object Request Broker Architecture

CSPF Constraint Shortest Path First

DiffServ Differentiated Services

DoS Denial of Service

DSCP DiffServ Code Point

DS Differentiated Services Field

ECMP Equal Cost Multi Path

EIGRP Enhanced Interior Gateway Protocol

FDM Frequency Division Multiplexing

GMPLS Generalised Multiprotocol Label Switching

IETF Internet Engineering Task Force

IGP Interior Gateway Protocol

IntServ Integrated Services

IOS Internet Operating System

IPTE IP traffic engineering

IS-IS Intermediate System - Intermediate System

ISDN Integrated Services Digital Network

IST Information Society Technologies

ITU International Telecommunication Union

ITU-T ITU Telecommunication Standardization Sector

LSP Label Switched Path

M-ISIS Multi Topology IS-IS

MESCAL Management of End-to-end Quality of ServiCe Across the Internet at Large

MIRA Minimum-Interference Routing Algorithm

MPLS Multiprotocol Label Switching

MT-ID Multi Topology ID

mt-OSPF Multi Topology OSPF

MT Multi Topology

NCS Network Configuration and Scheduling

OMG Object Modelling Group

OSI Open Systems Interconnection

OSPF Open Shortest Path First

PHB Per Hop Behaviour

PLC Packet Loss Concealment

PSTN Public Switched Telephony Network

QOSPF QoS-enabled Open Shortest Path First

QoS Quality of Service

RES-LSA Link Resource Advertisement

RPC Resource Provisioning Cycle

RRA Resource Reservation Advertisement

RSVP Resource reSerVation Protocol

RTP Real-time Transport Protocol

SNMP Simple Network Management Protocol

SPF Shortest Path First

SSH Secure Socket Shell

TCP Transmission Control Protocol

TDM Time Division Multiplexing

TEQUILA Traffic Engineering for Quality of Service in the Internet, at Large Scale

TE traffic engineering

TINA Telecommunication Information Network Architecture

TINA-C Telecommunication Information Network Architecture Consortium

TMN Telecommunications Management Network

ToS Type of Service

UDP User Datagram Protocol

VoD Video on Demand

VoIP Voice over IP

WSP Widest-Shortest Path

XML eXtensible Markup Language

Chapter 1

Introduction

1.1 The (not so) Simple IP Network

Whereas traditionally, IP networks are self-managed and low-maintenance, there is a growing need for fully integrated software management solutions that perform more than failure monitoring. The reason for this is that while IP networks are inherently simple to install and operate, this is only true as long as the services operated on these IP networks are non-critical and have relatively low requirements. Services such as web, ftp and email meet these requirements. None of them provide guarantees and all work reasonably well. However, email is a good example of how the non-critical perception of the Internet has changed already. Many business today are fully dependent on working email solutions and a 100% service is usually assumed. Suddenly, a service that was never conceived as reliable and robust has become mission critical to the world economy. A very interesting development in its own right, but given the positive experience that many non-technical people have had with the Internet, public opinion and hype have caused a frighteningly fast move towards all-IP solutions; fully integrated IP communications solutions with voice, email and even video sharing the same network infrastructure. With this comes about a change in the fundamental requirements from such IP networks and the simple features that made IP attractive in the first place have now become the very problems that need to be resolved. Because of

its simplicity, IP has no concept of end-to-end quality, no session awareness, no network-side control. As a result, capacity over-provisioning has become the solution for network backbone operators. The hope is that as long as the network operates at low utilisation, no problems will occur, since there is so much spare capacity that most of the conceivable problems can somehow be compensated for. However, despite the operators firm assurances that over-provisioning is a valid long-term solution, the networking community has recognised that more sophisticated techniques for traffic engineering are required. Techniques such as Multiprotocol Label Switching (MPLS) essentially replace IP with an ATM like solution and mark the other end of the spectrum. Either way, it is clear that the simple IP network is no longer simple and that solutions are required that can provide a cost effective long-term solution. Such a solution should be able to take into account existing techniques and have the ability to integrate these as well as up and coming new solutions.

1.2 Motivation

The push towards integrated network solutions capable of carrying all forms of network service on a single IP infrastructure is the core motivation of this thesis. These unified networks carry all types of traffic and are deployed much closer to the customers home than before. Whereas before, IP, PSTN and even TV connections were treated separately, carried to the home through separate carrier technologies, the new unified network infrastructures carry all services over IP directly to the local exchanges from where it is distributed to the home. Compared to current networks, the resulting network has the potential for a much simplified topology and is therefore easier to maintain and operate. User end advantages are claimed to include service portability, unified billing and a more integrated service experience. In addition to the major service types known today, future - yet undefined - services that arise with new requirements are likely to be carried over the same infrastructure. The lack of any service specific design criteria means that IP networks can

be adapted, while the only relatively recently highly acclaimed ISDN has already failed to accommodate developments such as higher data rates or the "always on" paradigm.

The success of multimedia applications on the Internet in the recent past also indicates the trend towards multiservice networking, this time from the user perspective. And despite the Internet's principal design as a simple datagram service, many performance sensitive applications now run quite well by adapting to the variations in network conditions [5]. But while applications have changed dramatically during the short existence of the Internet, network layer functionality has evolved very little. There are two reasons for this: the maturity of the technology (it works) and the inertia caused by the need to potentially update *every* router of the Internet to accommodate a new feature that may or may not be successful. Thus, while large efforts have been made by the research community to add features to the Internet that are mission critical to real time services not many of them have been deployed.

Fortz and Thorup [6] make the point that functionality that is subject to frequent change, such as path selection algorithms and policies controlling those algorithms, should not be located in the network layer because of this inertia. By contrast, if the functionality was located in a more centralised software management layer it could be modified more easily to represent changes in business objectives as well as improvements in technology without the need for updates to router operating software.

The approach presented in this thesis is based on building service differentiation on top of distributed routing protocols like OSPF. This is motivated by the hope that "traditional" IP routing techniques can be used to build future multiservice networks and that more recently proposed mechanisms like MPLS, which remove a lot of the attractiveness of IP networking are not required – at least not in a large scale deployment.

The approach is a combination of conventional distributed routing and management based traffic engineering functionality. There are some good arguments for splitting the functionality in such a way: the most important advantage is that the approach bares the opportunity for graceful migration that causes little disruption to existing layer 3 operations. With the exception of the use of multi topology routing which only exists in draft standards, most functionality is added at the management layer. Equally important is that any complexity introduced by Quality of Service (QoS) awareness remains outside the network layer, leaving the distributed best-effort Internet untouched. Moving the complexity caused by QoS awareness into the management plane also has the advantage of avoiding the large increase in state information in the network.

1.3 Thesis Objectives

The thesis objectives give a broad guide to the problems that are addressed in the thesis. While the work done can be described by these objectives, it does not exhaustively cover each objective.

1. Identify key problems and requirements for engineering a multiservice IP network.

2. Propose and evaluate a traffic engineering (TE) scheme based on OSPF or IS-IS that can be used to for routing differentiation on intra-domain networks.

3. Show how the proposed TE schemes can be used as a tool for managing multiservice networks.

1.4 Scope and Assumptions

The work of this thesis combines ideas from the following areas:

1. Traffic Engineering of intra-domain networks with the aid of traditional mechanisms, i.e. without using techniques such as MPLS.

2. IP network management for managing intra-domain networks, specifically those parts that concern intra-domain traffic engineering.

Some assumptions and limitations were made to create boundaries for the problem area:

1. Investigations have been limited to the routing level, no packet level simulation has been performed to validate any algorithms or assumptions. While packet level simulation can provide an extra level of confidence, the work done here is an investigation of routing schemes. Packet level simulations require a large number of additional assumptions, for example for prioritised queuing that can lead to inaccurate results unless ample time is invested into validating assumptions.

2. The study does not compare the numerous algorithms for link weight optimisation based on Fortz and Thorup, since they do not allow constrained optimisation and would thus need to be extended in order to be comparable.

3. While many areas of network management are touched, areas such as failure detection and management or traffic monitoring and forecasting are not discussed in any detail.

4. In order to perform link weight optimisation, a traffic matrix providing ingress and egress routers as well as traffic amount is assumed to be available. For the purposes of simulation, these matrices were generated, but while the techniques for retrieving such a matrix from a real network are discussed briefly, no exhaustive study has been done.

1.5 Novelty

The main contributions to the research field made by the author are presented in chapters 3, 4 and 5. Chapter 3 and 4 describe the IPTE traffic engineering technique based on the work done by Bernhard Fortz. The

author has built on the method described in the paper [6] and extended the approach to take multiple constraints into account, for example to enable QoS. He has extended the approach to be multi-topology aware and introduced the concept of using the cost function for different optimisation constraints for routing planes. He has implemented simulators to carry out this work prior to the release of the TOTEM toolbox (Bamatraf and Othman [7]), which was consequently not used for this work. In the meantime, several developments have taken place that have produced similar results, k-set traffic engineering Wang et al. [8] and MT-OSPF Psenak et al. [9] have been released, however as can be seen, a paper published by the author in LCS2004 [Gri04] pre-date these events. These new developments have been documented in the text for the purpose of providing an up to date view of the research field.

Chapter 5 of the thesis discusses the link weight setting algorithm in a network management setting. This chapter introduces some ideas developed by the author during the course of the MESCAL project, which is briefly described in chapter 2 section 2.7.4. The ideas presented in chapter 5 on intra-domain TE management are essentially novel, with some inspiration from project work.

1.6 Reading Guide

1.6.1 Organisation

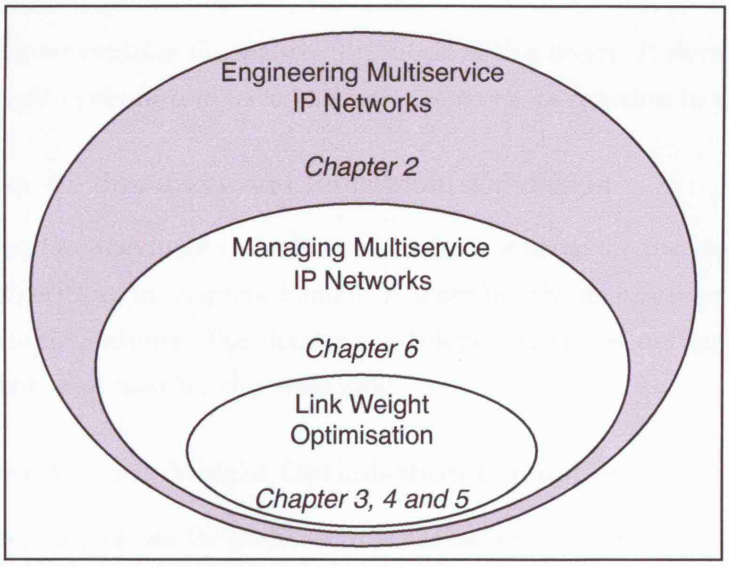


Figure 1.1: Organisation of Main Thesis Chapters

Figure 1.1 presents a graphical outline of the most important thesis chapters. Chapter 2 reviews literature and introduces some essential ideas. Chapter 3 describes the main contribution of the thesis, the link weight optimisation algorithm. Chapter 4 and 5 present the simulation methodology and results for the algorithms developed in chapter 3. Chapter 6 reintegrates the link weight optimisation with the bigger picture that was outlined in chapter 2.

1.6.2 Chapter Descriptions

Chapter 1 - Introduction

Chapter 2 - Engineering Multiservice IP Networks

This chapter introduces the most important concepts and defines how they are used in this thesis. It also serves as a review of recent work in this area of research where this is found to be relevant. Because of

the convoluted nature of the concepts introduced in this section, there is some degree of forward referencing in this chapter.

Chapter 3 - Link Weight Optimisation

This chapter contains the main contribution of this thesis. It develops the link weight optimisation technique and discusses its function in detail.

Chapter 4 - Simulator and Functional Validation

This chapter introduces the software simulator written for the evaluation of the algorithms in chapters 3 and 6. It describes the architecture as well as inputs and outputs. The chapter also briefly introduces any supporting tools that were used for the evaluation.

Chapter 5 - Link Weight Optimisation Evaluation

This chapter presents the evaluation of the link weight optimisation scheme from chapter 3. It discusses the simulation model that was used for validation of the technique is described and simulation results of for several scenarios are presented.

Chapter 6 - Managing Multiservice IP Networks

The fifth chapter combines the ideas of the previous three chapters to form a framework for managing the link weight optimisation in an operational network environment. The chapter shows how link weight optimisation can be used as a flexible tool forming part of a network management system working alongside other mechanisms. It further extends the concepts of chapter 3, based on the additional requirements placed by the management system.

Chapter 7 - Discussion and Conclusion

The discussion and conclusion chapter is a forum for surrounding discussion outside the scope of individual chapters. It reviews the scope of the thesis and its achievements and discusses the additional topics that were

not covered. Finally a summary of the main achievements and discoveries is presented as well as a list of personal achievements the author gained during the course of the work.

Appendix A - The Authors Role in Research Projects

Appendix A is a short discussion on the authors role in research projects he took part during the completion of his PhD. This is a clarification of the correlation of work in projects and thesis.

Publications

The authors publications during the course of the PhD.

Technical Reports

A list of project technical reports to which the author contributed during the course of the PhD. This relates to Appendix B, where the work is briefly discussed.

References

Chapter 2

Engineering Multiservice IP Networks

2.1 Introduction

The intention of this chapter is to give an overview of the requirements that are placed on the IP network infrastructure in order to build "better-than best-effort" services as well as giving a summary of the recent developments in the field. It should identify and outline the framework within which the authors work is placed. By no means is it intended to fully describe the operation of routing protocols, traffic engineering or management systems; extensive discussions on all these topics are widely available and extensively referenced. There are however, large variations in the interpretations of terms like QoS, TE, routing and network planning. The following text describes and defines terms as they will be used throughout the thesis.

2.2 Multiservice IP Networks - Definition and Requirements

Recent years have seen a push towards integrated network solutions capable of carrying all forms of network service on a single IP infrastructure.

Companies such as British Telecom with its "BT's 21st Century Network" initiative [10] and France Telecom with NExT [11] have begun transformations of their network cores alongside expensive marketing campaigns. These unified network cores carry all types of traffic and are deployed much closer to the customers home than before. Whereas before, IP was carried over ATM, leased lines and other technologies, while at the same time PSTN connections were treated separately, the new unified network infrastructures carry all services over IP directly to the local exchanges from where it is distributed to the home. Compared to current networks, the resulting network has a much simplified topology, which is easier to maintain and operate. [12][1] End user advantages are claimed to include service portability, unified billing and a more integrated service experience. In addition to the major service types known today, future - yet undefined - services that arise with new requirements are likely to be carried over the same infrastructure. The lack of any service specific design criteria means that IP networks can be adapted, while the only relatively recently highly acclaimed ISDN has already failed to accommodate recent developments such as higher data rates or the "always on" paradigm. In order to carry all types of voice and data services, a true multiservice network needs to be configurable to carry traffic with varied QoS requirements:

> "Multiservice networks provide more than one distinct communications service type over the same physical infrastructure. Multiservice implies not only the existence of multiple traffic types within the network, but also the ability of a single network to support all of these applications without compromising QoS for any of them."
>
> *Robert Wood [13]*

[1] Information about BT's 21st century network was largely taken from a presentation by Matt Beal, the director BT Wholesale at the time of writing. The reference contains a web link to presentations on BT's 21cn network website.

The quote summarises multiservice networks in few words and reflects the discussion above. A network that can support more than one service has to be constructed of small building blocks, which can be assembled, exchanged and redesigned to form the complex structure that is the face of the network today. It cannot be a rigid structure, simply because future requirements cannot be foreseen.

IP was not designed to be a building block for a multiservice network, but it does satisfy the criterion of being simple and non-restricting. IP is a datagram service with no delivery guarantees, other blocks towards building a multiservice network have been proposed over the years, but assembling these components is a complex endeavour that has to be carefully planned and managed in order to be successful. The remainder of this chapter is concerned with outlining the problems and then some of the current best practises addressing the problems of turning an IP network into a multiservice network.

2.3 Quality of Service

> "Obviously some things are better than others... but what's the "betterness"?... so round and round you go, spinning mental wheels and nowhere finding anyplace to get traction. What the hell is Quality? What *is* it?"
>
> *Robert M. Pirsig [14]*

2.3.1 Definition

Quality of Service is the key factor to turning an IP network into a multiservice network. The ability to provide service guarantees to voice calls as well as more advanced services is the difference between a good feature of the Internet and a useful reliable service that becomes ubiquitous and an indiscernible part of everyday life.

Adaptive audio and video codeces like RealVideo [15] that can tolerate variable bit-rates, packet-loss (Packet Loss Concealment (PLC) [16]) and jitter (jitter buffer [17]) are used in streaming audio/video applications like Real Player [18] and Microsoft Media Player [19]. Packet transport protocols that do not perform TCP [20] congestion control, like RTP over UDP (RFC1880 [21]) avoid the stalls in data transfer rate. Yet the calls for quality of service are justified, if the Internet is to replace, rather than complement any of the existing telephony and television services.

The term "quality of service" is often used as a buzzword for many different technologies and services that hope to achieve such guarantees. Arriving at a meaningful definition of "quality of service" for this thesis proved rather more difficult task than first anticipated. Though often used to describe a missing property of the current infrastructure, a precise definition of the term and hence a guide to required network improvement is hard to find. The following two definitions are quotations of definitions found in the literature. The first is a general definition from the ITU X.902 standard: Reference Model of Open Distributed Processing, while a more multimedia service specific definition can be found in the IEEE CASCON paper: "Distributed multimedia applications and quality of service: a survey".

> "A set of quality requirements on the collective behaviour of one or more objects."
>
> *ITU X.902 [22]*

> "...those technical and other parameters of a distributed multimedia system, which influence the presentation of multimedia data to the user, and in general the users general satisfaction with the application."
>
> *Vogel et al. [23]*

From these definitions it follows, that quality of service has a twofold meaning. It is a collective term for all technical parameters of a communications channel that are related to the quality of the transmission. For an IP (or any other packet-) network, such parameters are delay, jitter, packet loss, uptime, etc. It also follows from the second definition that QoS is the particular selection of the parameters that is important in the context of the application: the subjective rating of the quality. This is a more profound definition, as the subjective view of one and the same service may vary from one individual to another. In this subjective sense of the word, quality of service is a top down approach for defining the network parameters required to satisfy the majority of individuals. These network parameters differ between services. For example low delay for real time applications and low packet loss for data transfer. For voice calls it also means reasonably fast session setup time and high reliability. The network parameters for achieving QoS also vary between groups of individuals as discussed by Bouch and Sasse [24], where a number of recommendations (service predictability, feedback to user, etc.) are made for the design of network QoS, and it is concluded that it is more important to the user to receive the expected amount of quality that enables them to perform the chosen task than to receive some objective levels of quality (i.e. 1Mb/s as opposed to 2Mb/s). When assuming the subjective view of quality of service, the engineering task is essentially the appropriate selection of relevant network performance parameters for each application, based on a working business model, as well as the corresponding enforcement of these parameters in the network. The selection is part of the planning phase of the service and should provide sets of parameters aggregated in "classes of service" that can be implemented and enforced in the network.

2.3.2 Quality of Service in Context

In the circuit switched network built for voice telephony, QoS parameters were chosen according to the requirements for voice calls. These parameters became the fundamental design requirements for every aspect of the

network, the protocols and the physical connections. In the operational network, care has to be taken mainly to prevent call blocking, the effect that is caused by in incorrect balance between subscriptions and network capacity. Much simplified, call blocking occurs when no physical circuit is available between two subscribers that wish to make a call. This together with long-haul delays is one of the few forms of degradation to QoS known to the circuit switched network, since a call, once connected, does not usually worsen in quality (except of course, network equipment failure, which can create any number of effects and also in mobile networks where the air interface also causes failures and interference).

By contrast, an IP packet network does not have planned, fixed capacity circuits, but rather a best-effort packet delivery policy. The network is designed to attempt to deliver as many packets as possible even if its nodes and links are not constantly operational. This does not mean that there is no quality of service, just that its parameters were chosen to be different. In his Internet history, Hauben [25] states that the purpose of the ARPAnet was to provide resilient data communication over telephony networks. Its purpose was not to provide high data rates or in-order packet delivery. Following this design criteria, the ARPAnet's successor, still based largely on the same protocols, does not prioritise or police packets and it is therefore not straight forward to facilitate and maintain high quality voice calls on any IP network. The fundamental differences between circuit switched networks and routed packet networks are out of scope of this discussion, but the resulting implications and techniques for overcoming these implications for QoS have to be discussed briefly.

In principle, it is relatively easy to provide a low delay, in-order packet stream over an IP network. This can be done in controlled environments such as company networks, where the volume of traffic is known as well as the capacities of each route; for example Fraleigh et al. [26] proposed that only 15% of over-provisioning above the average data rate is required on highly aggregated core networks. But even this may not be

enough, the results presented by Choi et al. [27] give evidence that even the over-provisioned core network gives rise to occasional high delays, and hence calling for more control over traffic than over-provisioning can offer. If this is the case or if general capacity over-provisioning is not possible, admission control has to be introduced in order to block all traffic (preferably on a per service level, i.e. per call or video stream) that exceeds the planned capacity.

Many comparisons between over-provisioning and admission control exist. Generally, most service requirements can be met through sufficient over-provisioning and it is argued by Menth et al. [28] that both approaches require similar amounts of capacity for normal operation. However as argued by Houck and Meempat [29], in reality there is a limit on the amount of over-provisioning that can be applied at any one time and if demand surges beyond this limit (e.g. on special occasions), admission control is necessary to avoid degradation of the already running instances of services. The problem on a network such as the Internet is that many entities share control over the traffic that is injected and so an admission control system is difficult to implement. From a technical perspective it would require sharing extensive amounts of subscription information as well as dynamic information on utilisation and available capacity in a large distributed admission control system. The real problem could be the protection of interests however, as the shared information on customers and network details may be sensitive to the service providers, individuals or even countries involved. Even if admission control could be implemented, all traffic would be treated the same, leading to sub optimal resource usage, since all traffic has to be carried with the highest QoS standards, although only a fraction of the traffic actually requires this high quality treatment.

Lack of prioritisation is thus a problem that has to be addressed as well as admission control. Another problem is the lack of control over the routing process. Routing protocols based on shortest path algorithms ag-

gregate traffic towards a destination and the more aggregated the traffic flow becomes, the larger the chance of congestion becomes close to busy egress nodes. Aggregation happens naturally, as from any one node the routing protocol will choose only one path (the shortest) towards any one destination, thus when flows of traffic from different sources destined towards one destination meet at an intermediate node, they become aggregated. Once aggregated these flows never separate (unless techniques from section 2.5 are used), which limits traffic engineering choices. Several small flows are more easily balanced across a network than one large inseparable one.

Finally, there is currently no well established concept of constraint routing in OSPF implementations, in order to honour delay, jitter or bandwidth constraints. This comes as no surprise since such mechanisms require careful configuration. Constraint based routing will be discussed in detail in section 2.5.6.

2.3.3 Queuing

A major factor in understanding and controlling QoS on packet networks is the queuing delay incurred by packets at each router. It is tempting to naively model the queuing delay as an average value incurred by a packet each time it crosses a router. Complex arrival times, fluctuating utilisation across the network and variations in packet size however, make it impossible to simplify queuing in such a way. Complexity introduced by queueing gives rise to three quantities: packet queuing delay is the delay incurred by one packet as it travels across one or multiple queues, packet jitter is the variance of this delay across a sequence of packets and packet loss, which occurs if a queue reaches its maximum length and any further arriving packets are dropped. A large jitter value can lead to out of order packet arrival, where one packet "overtakes" another and arrives out of sequence with respect to other packets of the same stream (Welzl [30]).

Router queues are typically modeled as M/M/1 queues, mainly because they are relatively simple to model. M/M/1 assumes packet arrivals to follow a negative exponential distribution. The same is assumed for the server, i.e. one server with a negative exponential service distribution. While any variation in arrival time is related to the time it takes for a new packet to arrive a the queue, service time is related to the length of the packet. The M/M/1 queue gives rise to a delay of (2.1)

$$\text{mean time in system: } \frac{1}{\mu - \lambda} \tag{2.1}$$

where μ is the *service rate* and λ is the *arrival rate.* A plot of this function is shown in figure 2.1. It shows that single packet delay in the

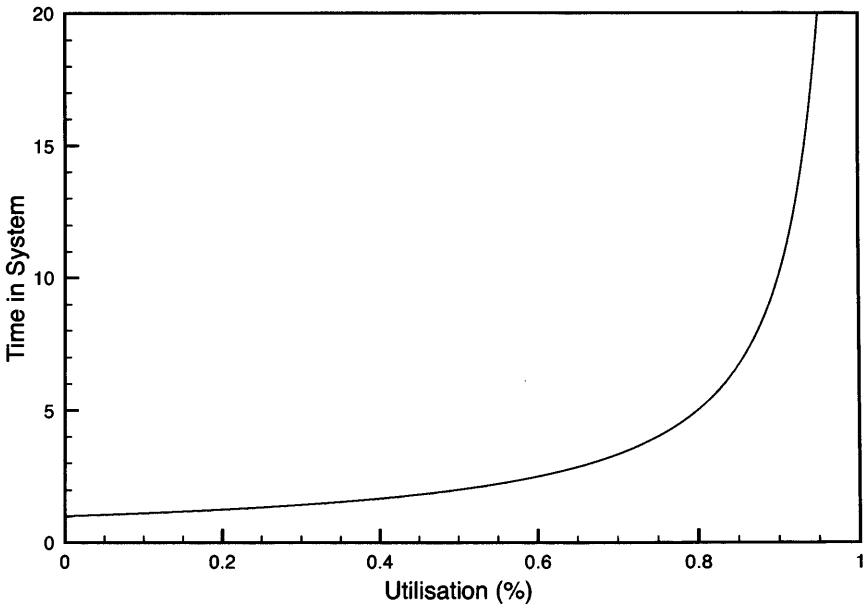


Figure 2.1: M/M/1 Queueing Delay with $\lambda = 1$

queue increases drastically as λ approaches μ and goes to show that unless all routers inside a network can be assumed to operate at low load, the resulting queueing delay effects can vary greatly and hence make any assumptions on delay and jitter difficult. Thus, queuing effects are an

important factor in controlling QoS for delay sensitive services (Asmussen [31]).

2.4 Network Engineering Techniques

2.4.1 Routing

The primary objective of routing is to discover paths between source and destination pairs across a network of nodes and links. Once paths are found, packets can be forwarded along these paths, given a valid destination address is supplied that each packet has to carry in its header. As a result of the design requirements placed by the ARPAnet project (which have since manifested themselves as the distributed philosophy of the Internet), route computation is always co-located with the forwarding nodes: routers. While distributed routing has many advantages in terms of resilience and robustness, ensuring the consistency of routing tables within each router becomes increasingly difficult with growing network size. It is therefore important to minimise the amount of information that has to be interchanged to keep up to date routing tables (Huitema [32]).

Most routing protocols today are designed to find shortest paths, the two predominant protocols are OSPF (RFC2328 [33]) and Intermediate System - Intermediate System (IS-IS) (ISO10589[2]). Shortest paths have the advantage of minimising the amount of network resources required per path as well as providing a crude attempt to providing "good quality" packet transmission (using the logic that shorter paths lead to quicker delivery). Thus both OSPF and IS-IS are based on a Shortest Path First (SPF) algorithm originating from the shortest path algorithm first described by Dijkstra [36].

[2]Since IS-IS is an ISO standard, its content is republished in RFC1142 for the Internet community [34]. RFC1195 [35] specifies IS-IS for the TCP/IP network.

Traditional routing takes the approach that the network is as simple a design as possible [37]. It lacks network side awareness of traffic conditions and congestion therefore has to be taken care at the network edges. However, general awareness of traffic conditions on the network side is a useful tool to help reduce congestion and it is has to be performed at least at high level by network engineers. Solutions can be categorised into *Network Planning or Provisioning* "putting capacity where demand is" and *Traffic Engineering* "putting traffic where capacity is". Network planning is generally used as a long term solution, whereas traffic engineering is a shorter term, ongoing optimisation of the available resources (Cheng et al. [38]).

2.4.2 Network Planning

The task of network planning is to ensure there is enough capacity in the network for future demands. This task is typically accomplished by predicting future network traffic and then providing network topology upgrades in order to serve the predictions. The quality of the results of a network planning effort thus depend heavily on the predictions of future demands and so the task of prediction is crucial (Cheng et al. [38]). In addition, the task of upgrading network equipment is usually disruptive to network operation as well as limited in flexibility (typically components have to be ordered and physically installed by engineers). It follows that network provisioning of this kind should be performed infrequently. Network planning is therefore typically performed on longer time scales. As a result of the long intervals and the coarseness of the provisioning (i.e. the installation of a new link), capacity is over provisioned just after a new installation has been made and rather limited just before re-provisioning has to take place. Also, network provisioning is not suitable for emergency reconfiguration of the network in case of failure (re-provisioning may be triggered by failures, but is hardly suitable as a short term solution).

More recently, with the advent of optical layer technologies, provisioning of network capacity is beginning to become more automatic and there-

fore faster [39]. Generalised Multiprotocol Label Switching (GMPLS) (RFC3471 [40] is the umbrella term for all optical MPLS technologies standardised by the Internet Engineering Task Force (IETF), whereas Automatic Switched Optical Network (ASON) (Larkin [41]) is the technology favoured by the International Telecommunication Union (ITU). Both technologies make it possible to allocate wavelengths (Frequency Division Multiplexing (FDM) or Lamda switching in the GMPLS context), time slots within wavelengths (Time Division Multiplexing (TDM)) and physical paths (space division multiplexing) on a dynamic base and thus enable short term[3] network provisioning through these lower layer technologies. Due to their fast provisioning capabilities these technologies, although traditionally network provisioning tools, can now potentially used within the time frame of TE mechanisms. Strictly speaking they are still network provisioning techniques, since they provide capacity that was previously not available. However, from the point of view of an optical network provider, they are traffic engineering tools, since they allow one to reconfigure the provisioning of existing capacity.

2.4.3 Traffic Engineering

The tasks of traffic engineering are opposite to those of network planning, here the network topology is assumed to be fixed and the traffic is rearranged to make better use of the existing topology. In this context *better* is typically the maximisation of profit to be gained from the network which, depending on the business model of the network operator, can be interpreted as serving as many customer demands as possible. To achieve this, the network operator might strive to balance the traffic on his network evenly so that it can accommodate additional load from new customers in different areas. However, the business model could have any number of other criteria, such as giving preference to certain customers over others and therefore allocating spare capacity to them. Also, in

[3]Clear information on "how short" is still hard to come by, as both ASON and GMPLS are still recent developments.

a QoS enabled network, preferences can be more differentiated and any number of requirements could be assigned to individual traffic trunks.

Mechanisms for traffic engineering *must* interfere with conventional SPF to increase the flexibility of traffic distribution across the network. It is proposed here that traffic engineering could be seen as an evolution of routing towards a more general form of path optimisation that takes into account not only the "selfish interests" of a single source destination pair, but a more general viewpoint of the whole network. When looked at in this way the importance of traffic engineering in a network naturally becomes larger with increasing network size, as a means for managing the increasingly complex traffic patterns, sometimes within confined capacity limits.

As with routing and forwarding before, traffic engineering consists of two instances: finding the most efficient distribution of traffic and configuring the forwarding behaviour of the nodes in the network. Because of this, there is no distinct separation between traffic engineering and advanced routing mechanisms. The following gives a summary of relevant techniques.

2.5 Summary of Traffic Engineering Techniques

2.5.1 DiffServ and IntServ

The Differentiated Services (DiffServ) framework (RFC2474 [42]) is an attempt by the IETF to provide packet queuing prioritisation inside the routers to address the problems discussed in section 2.3.3. DiffServ became the new approach to the QoS problem after it became apparent that Integrated Services (IntServ) (RFC1633 [43]) was not as well received by operators and vendors, because of scalability problems caused by the per flow bandwidth reservation (for example discussed by Welzl [30]).

DiffServ uses the 8-bit Type of Service (ToS) field in the IP header as Differentiated Services Field (DS) with 6 bits reserved for the DiffServ Code Point (DSCP) and two unused bits. Packets have to be marked individually at the ingress to the DiffServ network in a process known as traffic conditioning. Each DSCP is mapped to a Per Hop Behaviour (PHB), which provides different levels of prioritisation in router queues with well known behaviour inside a domain. PHBs can be used to map packets of similar type onto the same queue to reduce jitter, delay is reduced for the high priority queues simply by preferential treatment of packets in these queues.

However, as argued by Goderis et al. [44] the aggregation of traffic into different queuing behaviours is not sufficient for providing better-than best-effort services. Nothing stops the different queues from becoming congested and nothing ensures that delay sensitive data is forwarded along the shortest available path. From comparison studies of DiffServ and IntServ, such as the one made by Cisco Systems [45] it can be seen that DiffServ avoids the "statefulness of IntServ by limiting itself to the definition of queuing behaviours for packets in individual components. There is no end-to-end concept as in IntServ.

In contrast to IntServ, DiffServ therefore provides a building block for QoS delivery, rather than a complete solution[4]. Of course the lacking end-to-end management has to be provided by other means, in order for DiffServ QoS solutions to become complete. RFC3175 [46] attempted to reintroduce end-to-end bandwidth reservation in an aggregated form suitable for DiffServ, however, for true control of the path it is unavoidable to add at least two more functional components to the IP network, *Admission Control* to ensure that no more traffic enters the network than can be supported, and load balancing to ensure that no links are overloaded

[4]Cisco Systems have integrated DiffServ support into their IOS router operating system in 2001. It is speculated here that this incremental approach has a larger chance of market success, as component vendors and network operators are likely to prefer incremental, quickly implementable solutions to more radical changes.

in the planned zone of operation. This is also discussed by Goderis et al. [44].

2.5.2 ECMP

As discussed previously in section 2.3.2, SPF algorithms have a tendency to aggregate traffic and thus potentially creating large unbalanced flows of traffic even if another less loaded path is available. This has been recognised by the community and as a result, Equal Cost Multi Path (ECMP) was developed, which is now supported by several routers (e.g. Cisco [47]). ECMP is a technique that allows the equal splitting of traffic at nodes where multiple shortest paths exist to a destination. Several mechanisms are offered to perform the splitting, which fall into two categories: (1) round robin per packet splitting and (2) based on a hash function. Hash functions can be applied to source address, group and source address, etc. to ensure that individual traffic flows are not split as in the round robin case.

In addition to the basic use of ECMP for load splitting, it can also be used for resilience as shown by Iselt et al. [48]. In the same paper the authors also mention that ECMP requires careful engineering of equal length paths at points of network congestion, which might be a limitation. There are also concerns about introducing jitter, where individual Transmission Control Protocol (TCP) flows are split as pointed out by Awduche et al. [49]. Finally, Kandula et al. [50] point out that since ECMP is not aware of congestion, it can cause overloading of links by equally splitting traffic onto already congested links. The authors thus present a dynamic means of performing splitting.

2.5.3 Dynamic Routing

Neither OSPF nor IS-IS have congestion awareness features (as per RFC's 2328 and 1142) and since the shortest path for most source-destination

pairs cross the most highly connected nodes in the centre of the network, congestion can occur in badly designed networks.

Traditionally the problem of congestion has been addressed in the transport protocol at the network edges by TCP (Allman et al. [51]). The reason for keeping routing protocols unaware of congestion is the path instability introduced into the network, when introducing congestion awareness into the routing algorithm (Huitema [32]). Bertsekas [52] argues that congestion aware routing exhibits complex behaviour that need to be taken into account during the algorithm design. For instance, once a point of congestion has been detected, a careful balance has to be struck in order to redirect traffic so as not to create congestion elsewhere in the network and possibly at the same time, underutilising the previous point of congestion. If this balance is not found, consecutive routing table updates may switch back and forth between sub-optimal configurations, each causing congestion in the network. Worse, the resulting route flapping could potentially render the network unstable to a point where it becomes unpredictable as demonstrated by Wang and Crowcroft [53] as well as Khanna and Zinky [54].

2.5.4 MT-OSPF and M-ISIS

Awareness to the DSCP or ToS byte were originally included in the OSPF standard (RFC1583 [55]). It was later officially dropped in RFC2328 [33] and with it, any ability to differentiate routing based on DSCP[5]. More recently some Internet drafts have appeared that introduce "multi topology routing" or mt-OSPF Psenak et al. [9] with the corresponding DSCP's successor being the more friendly termed Multi Topology ID (MT-ID).

Multi topology routing is essentially the same idea as ToS byte routing, but it is no longer linked with DiffServ or other QoS routing. Each multi-

[5]While support was dropped, the RFC retained a mention of the feature for reasons of backward compatibility.

topology is a logically separate instance of OSPF with a separate "routing plane" containing its own routing and forwarding table. Differentiated routing between the planes can be achieved by manipulating the link metrics of each plane. The MT-ID in the packet header field determines which topology a packet is routed over. An MT-ID can be set at the network ingress point or it can be a more commonly known inter-provider service differentiation descriptor such as defined by Griffin et al. [56]. How the MT-ID is set is not specified by the draft.

A similar draft was written for IS-IS by Przygienda et al. [57], termed M-ISIS. Although this draft has also expired, Cisco Systems [58] have included an implementation into their Internet Operating System (IOS). As of 2007, Cisco Systems [59] have generalised the multi topology support to routing protocols including Border Gateway Protocol (BGP), IS-IS and OSPF. In order to conform with the most recent nomenclature, MT-ID will be used in place of DSCP in this and the following chapters. Multi-topology routing will be covered in more detail in the next chapter.

2.5.5 MPLS

Multiprotocol Label Switching (RFC3031 [60]) reintroduces some features of Asynchronous Transfer Mode (ATM) [61] into the IP world. It is a mechanism that provides a means for configuring packet forwarding at network nodes without using an Interior Gateway Protocol (IGP) routing protocol. As such, MPLS introduces a clean separation of control and forwarding planes. By making this separation, core network nodes running MPLS do not have to compute routing information. Instead, traffic is forwarded on preconfigured paths by means of a label lookup at each node, giving the network operator a choice of many different types of control plane components to configure Label Switched Paths (LSPs). By itself MPLS is not QoS aware, however, it can be used in combination with prioritised queuing such as DiffServ PHBs (defined in RFC3270 [62]) and appropriate control plane components to become a QoS enabler.

The main argument for MPLS was initially the reduction of computing requirements at the network nodes, thus allowing the construction of faster networks. This argument has faded with increasing computing power, but MPLS has since become more widely used by network operators as a means to explicitly route traffic across IP networks that are also operating an IGP. These "tunnels" have allowed Internet access providers to separate bandwidth used for Internet access from that reserved for added value services, such as Voice over IP (VoIP). Strategies for combining MPLS and IGP in a more rigorously optimised fashion are described by Ben-Ameur et al. [63], who proposes that only few tunnels are required to create an effective TE solution. Ben-Ameur et al. also propose a similar method to use with ECMP, but note that MPLS is a more powerful mechanism that can address multiple constraints, whereas ECMP is more limited (as already discussed here).

While MPLS has the potential to be more flexible than IP/IGPs for service differentiation and as a QoS service building block, it has some inherent problems. If the control plane is no longer distributed, IP networks will loose one of their most compelling advantages: their lack of a single point of failure and their inherent robustness to node and link failures. While many such features have been added to MPLS (including RFCs for RSVP extensions [64], fast re-route [65] and fault tolerance guidelines [66] to name a few), the resultant design is not as "natural and clean" as that of IP/IGP solutions. There is also a concern of scalability, since all paths have to be explicitly configured at each node, adding state information at each hop, however, node memory should not cause any concern considering the recent advances in computing.

The lack of inherent properties in MPLS combined with the potentially large number of LSPs in an operational network may also be cause for concern. From the point of view of a single node, a large MPLS configuration would look like a maze. As such, the complexity of managing a large MPLS network in the face of a failure or a personnel change might

leave it open to miss-configuration with the potential of causing large scale failures with long recovery periods.

2.5.6 Constraint Shortest Path Algorithms

Apostolopoulos et al. [67] study the complexity caused by QoS aware routing algorithms, both in terms of increase in protocol overhead as well as computational complexity. They propose policies that limit the amount of overhead, by introducing update timers and by regulating the sensitivity of updates. The also study strategies for reducing cost of routing computation.

Constraint Shortest Path First (CSPF) is a modification to the Dijkstra SPF algorithm that allows shortest path selection according to some traffic engineering constraints, such as bandwidth and hop count. The CSPF algorithm has been associated with both OSPF and MPLS. CSPF is currently used mainly for minor routing constraints, such as exclusion of particular nodes or links. Requirements for QoS based routing are outlined in RFC2386 [68].

A large number of constraint routing algorithms exist, many of them were formulated to perform QoS routing, some are addressing network failure states. The following is a selection of unicast algorithms.

QoS extensions to OSPF were presented by by Guerin et al. [69] (a draft was also released Zhang et al. [70]). Termed QoS-enabled Open Shortest Path First (QOSPF), it was proposed in 1996 and defines two messages Link Resource Advertisement (RES-LSA) and Resource Reservation Advertisement (RRA). Using the RES-LSA, available resources are advertised in the network. Updates occur when link bandwidth consumption changes as well as when new nodes are added to the network. The purpose of the resource reservation advertisement is to describe how much capacity each router has reserved for a particular flow. The routing algo-

rithm developed for QOSPF is Widest-Shortest Path (WSP), as its name suggests it chooses the link with the largest amount of residual capacity.

A further development to this was presented by Kar et al. [71] in Minimum-Interference Routing Algorithm (MIRA). The MIRA algorithm takes into account what these authors call interference with other demand pairs. Any WSP will inadvertently affect traffic on other paths when it is created, thus MIRA tries to find paths that have minimal effect on the total flow capacity of the network. Several other algorithms fall into the category of minimum interference: Extensions to MIRA were proposed by Bin Wang; Xu Su; Chen [72] as well as Atteo et al. [73]. The SAMCRA algorithm Mieghem and Kuipers [74] is also built on TAMCRA by de Neve and van Mieghem [75] and proposes further minimum interference routing algorithms. While TAMCRA was designed under ATM criteria, SAMCRA is a new approach for IP networks.

Cui et al. [76] present a multi-constraint QoS aware algorithm called PMCP for intra-domain routing. The proposed algorithm is router based and builds one routing table based on a number of constraints.

The algorithms presented by Korkmaz and Krunz [77] and Zheng et al. [78] are a combination of pre-computed and on demand routing and claimed to be faster than other QoS aware routing algorithms.

A extensive overview of QoS aware routing algorithms was published by Chen and Nahrstedt [79] and a second more up to date overview was published by Zheng et al. [80]. These provide a more complete picture than the one presented here.

2.5.7 OSPF Link Weight Manipulation

Traffic engineering in larger networks with more complex organization of traffic flows calls for finer grain control of traffic forwarding than IP routing and forwarding with its "source invariance" is able to provide. Xiao

and Ni [81] highlight the benefits of having control at traffic trunk granularity. Mechanisms for implementing more finely grained traffic engineering solutions like ATM, MPLS or IntServ that were discussed before have the same disadvantages: more state information has to be kept in each node than in IP/IGP routing in order to "memorise" each path, causing scalability and manageability problems especially in large networks.

OSPF and IS-IS provide for a crude traffic engineering mechanism by means of link weight manipulation. Originally conceived to find the "shortest" path these weights can be manipulated to make some paths appear shorter to the routing algorithm than others, hence giving the network operator a means for interfering with the routing process. The standard recommendation for setting link metrics is the well known inverse capacity setting as recommended in the Cisco ISO manual [82]. The proportionality of capacity to "path length" causes routing to prefer larger capacity links, building on the assumption that those links are found in the network core.

More recently research in IGP link weight manipulation has tried to avoid the per flow or per aggregate flow treatment of traffic and perform traffic engineering simply by cleverly setting the IGP link weights. Any traffic pattern that can be routed with link weights can also be routed with MPLS, therefore the goal of link weight manipulation is to perform close to mechanisms like MPLS, not better. The approach differs fundamentally from CSPF or QOSPF, as it does not modify any part of OSPF itself.

The first mention of OSPF link weight optimisation was published by Fortz and Thorup [6] in 2000. This marks the first attempt to use a heuristic algorithm for extending the use of link metrics beyond the crude TE through inverse capacity settings. While the problem of distributing the flow across the network can be formulated as a linear program, solving it with the constraints of SPF routing is NP-hard. The process developed

by Fortz and Thorup therefore uses a search heuristic to obtain a set of link metrics. The heuristic searches a simulated version of the operational network that uses estimated or measured traffic demands. These metrics should then perform similarly in the operational network, depending on the quality of the network simulation. Of course this means that in absence of regular re-optimisation, the network will "just" be an OSPF routed network.

In the paper Fortz and Thorup [6] describe the heuristic algorithm that allows the calculation of a total cost for network link load and hence identifies expensive links that do not fit the objectives which are defined in a cost function. The algorithm attempts to reduce the load on these expensive links by conducting a heuristic search of the links neighbourhood and subsequently readjusting a link weight to shift some of the load onto a suitable neighbour. The process iterates until a good solution is found. The details of the approach will be described in the next chapter.

A number of papers have been published extending or modifying the approach of Fortz and Thorup [6]. Work by Ericsson et al. [83] is applying a more generic genetic search algorithm instead of the local search algorithm applied in the original work. Results were shown to be similar to those of the original paper. Improvements to the original heuristic scheme were made by further extending the genetic algorithms. Buriol et al. [84] reintroduced a local search procedure in addition to the genetic heuristics to create a hybrid (memetic) algorithm. Algorithms presented by Ye et al. [85] claim to have achieved improvements of 50-90% in convergence time over the heuristics of Fortz and Thorup [6].

Research by Riedl [86] uses bandwidth and delay metrics for optimisation of multi-metric routing protocols such as the Enhanced Interior Gateway Protocol (EIGRP) by Systems [87]. However, the metrics are used purely as further traffic discriminators for traffic engineering optimisation purposes, explicit QoS guarantees for the flows is not intended.

Fortz and Thorup themselves proposed additions to their work. In [88], they describe a method for optimisation with few weight changes. This method limits the number of changes the algorithm is allowed to make in order to minimise disruption to the network operations. Employing a proactive rather than the reactive approach from [88], a further extension to the original algorithms was proposed in [89], addressing link failures by taking the failure states of most critical links into account when minimising the optimisation cost function.

A toolbox for performing traffic engineering simulation was released by Bamatraf and Othman [7] as part of the TOTEM project. This toolbox contains the link weight optimisation algorithms by Fortz and Thorup as well as the c-BGP simulator developed by Quoitin and Uhlig [90]. The toolbox is open source and therefore lends itself well to customised simulation of link weight optimisation problems.

More recently, Wang et al. [8] have shown that the results of Fortz and Thorup can be improved if the traffic is spread across more than one routing topology. The k-set termed approach shows that 4 topologies are sufficient to distribute the traffic nearly as well as an optimal flow scenario that could only be routed with an explicit forwarding mechanism like MPLS.

Teixeira and Rexford [91] present a framework for managing routing disruptions in IP networks. It includes a taxonomy of causes for internal and externally caused routing failures and strategies for using link weight optimisation and BGP policies for reducing the disruptions.

Sousa et al. [92] present a genetic algorithm for optimising IGP link weights with delay and congestion constraints. They use cost weighting to model the different constraints, but recognise the need for using a multi class approach for their future work in order to better address

the differing requirements of QoS delay and bandwidth classes in order to perform optimisation for multiple objectives.

Although not strictly focusing on link weight optimisation, Bagula [93] propose a genetic algorithm for hybrid MPLS/IGP networking in order to perform load balancing. They conclude that only a few LSPs are required to run in parallel with the IGP to reduce the network load by more than 30% on their test network. This shows that a large improvement over IGP routing can be achieved, if a few carefully selected traffic demands are routed independently from other traffic. However, it is hypothesised here that the success of the approach depends largly on the existence a few traffic demands that cause imbalance if routed over the shortest path.

For completeness, further studies were made in Piro et al. [94], Pettersson et al. [95], Srivastava et al. [96], Retvari and Cinkler [97], Lorenz et al. [98]. These studies all discuss the link weight optimisation problem of Fortz and Thorup and extend or modify the algorithm to improve the load balancing properties.

2.6 Traffic Matrix Prediction

Most forms of TE as well as network planning introduced in the previous sections require information about the traffic that is currently flowing through the network. In its simplest form a traffic matrix contains the volume of traffic between ingress and egress nodes over a given time interval. Longer term configuration tasks also require predictions about the future demands on the network, so that spare capacity can be introduced where it is required. Unfortunately while the quality of network engineering outcomes depends heavily on the accuracy of the traffic matrix, the generation of matrices (for current or future demands) is not a simple task. The research presented by Roughan et al. [99] gives some insight into the differences between using real and estimated traffic matrices, highlighting that when using predicted traffic matrices, it is important to

use robust forms of traffic engineering so as to give a degree of flexibility to any errors in the matrix. They show that OSPF link weight optimisation is particularly suited for predicted traffic matrices, while MPLS suffers because its static routes do not allow for a large error margin[6].

Techniques for generating traffic matrices can be broadly classified into those using flow level measurements (Feldmann et al. [100]) and those that derive the matrix from more general link load data (Medina et al. [101]). While a traffic matrix derived from per flow level data is potentially highly accurate, it requires techniques such as deep packet inspection or flow level inspection to collect the necessary information. Since these techniques are equipment intensive, they have a problem with scalability at reasonable cost. The link load data necessary for the techniques for deriving traffic matrices on the other hand can be retrieved from the routers via the Simple Network Management Protocol (SNMP) with very little overhead.

Methods for deriving traffic matrices can be further categorised into those using "gravity models" and those that combine gravity models with "network tomography" methods. Gravity models relate the total amount of traffic passing through each node in a pair to the distance between the two nodes: the amount of traffic between the pair increases proportionally to the total amount of traffic passing through each node and decreases proportionally with distance (Kowalski and Warfield [102]). Tomography (much simplified) is a technique for inferring a traffic matrix by using measurements from a subset of network nodes to re-construct the state in the whole the network (Vardy [103]). (Not to be confused other definitions of the term, such as "the science of inferring performance characteristics of the network interior by correlating sets of end-to-end measurements" [104].) Both tomography and gravity ideas are combined in several works, such as the one by Zhang et al. [105] to achieve better results.

[6]Of course MPLS is not a routing protocol and if a robust protocol were chosen to implement MPLS paths this problem would not exist. The statement about MPLS is based on its predominantly static means of implementation.

2.7 Network Management

2.7.1 Assembling the Multiservice Network

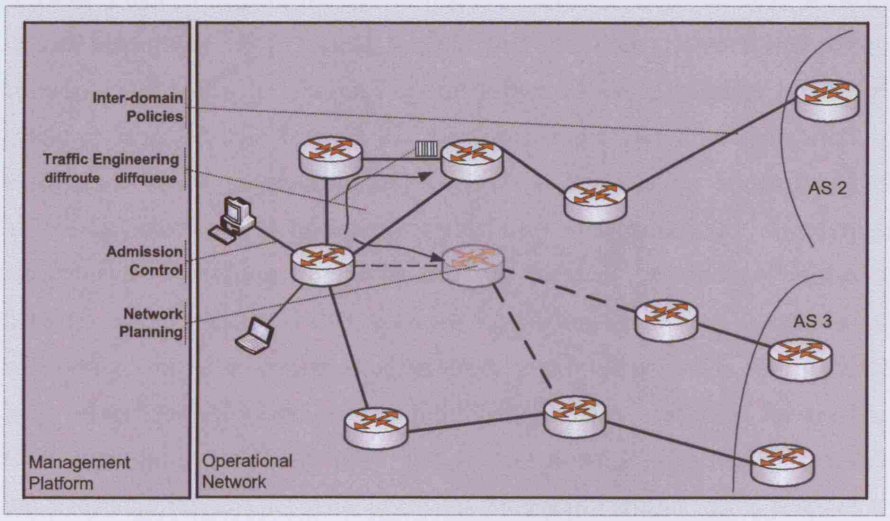


Figure 2.2: Critical Multiservice Management Functionality

The preceding discussion has shown the increase in complexity that is brought about by traffic engineering, admission control and other mechanisms to help support services with better-than-best-effort quality. While each of these systems work independently from one another, they are only effective at delivering end-to-end QoS if information is passed between them. For example the admission control system cannot decide by itself how much traffic it should allow into the network, since this information is dependent on the configuration of the traffic engineering component and the current utilisation of the network. The interdependency between the individual components might be one of the factors that is delaying the deployment of end-to-end QoS services, since it requires rethinking some of the fundamental "traditions" of IP networks: from the simplistic plug and play model towards a more rigorously managed network style.

In order to bring all the components for end-to-end QoS together, a controlling management component is needed that has the necessary over-

sight and authority to provide the "glue" between the various functions shown in figure 2.2 and configure them according to the operators overall policies. Such oversight already exists in today's networks in form of network operations engineers, but in a more complex QoS enabled network it may no longer be practical to control the whole network infrastructure by means of a few human operators who have control of the entire system. Instead, the task of low level oversight could be delegated to a software based network management. Such a system keeps track of all components and their inter-working as well as providing aggregated information to human operators in some form of "mediation" between network and operator. The network management system is concerned with events on time scales ranging from low level network operation to long term network planning. It influences routing and data forwarding though configuration of routers, but it does not take routing or forwarding decisions for the network and thus operates "offline" with respect to activity in the network. Likewise, it provides aggregated information to the human operators that allow them to quickly assess the situation and take important decisions.

Numerous management systems based on this layered design have been proposed. The first two standard architectures that have emerged are the Telecommunications Management Network (TMN) and the Telecommunication Information Network Architecture (TINA). Later architectures have largely built on either or both of these designs.

2.7.2 TMN and TINA

The TMN standard was first published in 1989 by the CCITT study group (which later became ITU-T). The latest revision of TMN (M.3010) was published in 1996 [1]. The standard introduces a 5-layer model to network management similar to the Open Systems Interconnection (OSI) 7-layer model and is depicted on the left hand side of figure 2.3. The main purpose of TMN was to provide a uniform framework for managing

increasingly large telecommunications networks. As such it is a heavy-weight standard, best suited for large networks. As in the OSI model, the purpose of the layers is to abstract the detailed information necessary to operate individual components and replace it with standard interfaces. These interfaces can in turn be operated by a higher layer control sys-

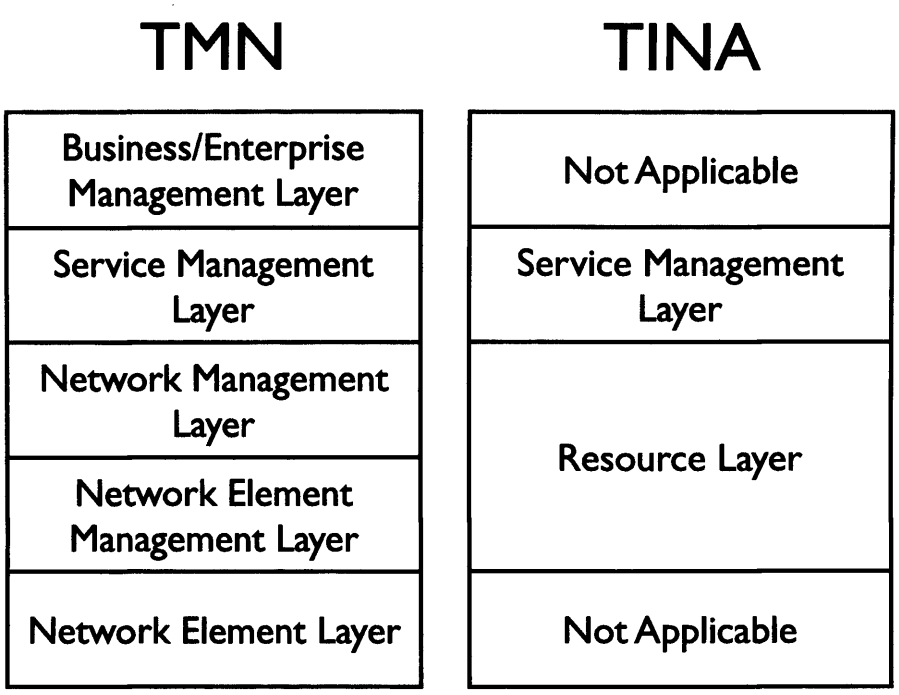


Figure 2.3: The TMN [1] and TINA [2] Management Architectures

tem, which in turn provides abstracted information to the next layer. This structure not only reduces the complexity of individual component design, it also allows for low layer technology to be replaced without having to update the whole system (Lewis [107]). The individual layers of TMN are defined as follows:

Business/Enterprise Management Layer This is concerned with level business objectives and long term planning. Another objective might be to extract information from the network to ascertain if business objectives have been met.

Service Management Layer This layer is concerned with translating the requirements of individual services to network requirements that can be addressed by the network management layer. It is also responsible for interaction with users and other service providers for billing and order processing.

Network Management Layer Concerned with bandwidth management, network monitoring and end-to-end control of congestion and other quality aspects. This layer requires extensive interaction between the different components in order to achieve this end-to-end control.

Network Element Management Layer This layer is concerned with management of individual network elements, routers, switches, bridges, etc. This is typically the operating software of the individual devices.

Network Element Layer Contains the actual network elements.

While TMN also included a service layer, its main focus was on the network management aspects. The Telecommunication Information Network Architecture Consortium (TINA-C) specified the TINA architecture with an explicit focus on the service layer in 2000 [2]. TINA collapses the network management into a single resource layer, in order to indicate an abstraction from the network, which is not directly considered. The TINA architecture is shown on the right hand side of figure 2.3. Consistent with the popularity of active networks at the time, TINA is based on intelligent networking and aims to provide a complete management solution for all types of service from service negotiation to tear-down (Lewis [107]). The architecture builds on a network of agents, which communicate to solve a particular task. A combination of such agents is involved in each service: a profile agent responsible for interaction with the user, a session agent manages the set up of the selected service and a resource agent allocates sufficient resources to the selected service (Lewis [107]).

2.7.3 Communication Between Components

Several protocols exist for the configuration between individual layers and components. For the configuration of network components in TMN, the reatively lightweight SNMP protocol was widely adopted. Communication between agents in TINA requires a more comprehensive language, such as Common Object Request Broker Architecture (CORBA) first defined by the Object Modelling Group (OMG) in 1989 [108].

More recently, routers have begun to support a variety of configuration protocols, depending on router operating system, for example such as specified in Cisco's eXtensible Markup Language (XML) Application Programming Interface (API) guides [109], which use CORBA, but also promise support for other mechanisms like Secure Socket Shell (SSH).

2.7.4 IST-TEQULA, IST-MESCAL and IST-AGAVE

Recent work on IP network management has produced architectures similar to the TMN framework, for example the TEQUILA architecture [110] for intra-domain management, the MESCAL architecture [3] for inter-domain management and the AGAVE architecture [111]. AGAVE[7], MESCAL[8] and the preceding TEQUILA[9] projects have followed a multilayer network management approach similar to that of TMN. Figure 2.4 shows the complex functional architecture of the MESCAL system.

[7]AGAVE - EU IST 5th Framework Project. Reference: IST-2005-27609, Start Date: 01-12-05, End Date: 31-05-08, Partners: Telefonica (co-ordinator), France Telecom R&D, Algonet, University College London, University of Surrey, Universite catholique de Louvain Official Website: http://www.ist-agave.org

[8]MESCAL - EU IST 5th Framework Project. Reference: IST-2001-37961, Start Date: 01-11-02, End Date: 30-04-05, Partners: France Telecom R&D (co-ordinator), Thales Research and Technology Limited, Algonet SA, University College London, University of Surrey, Official Website: http://www.mescal.org

[9]TEQUILA - EU IST 5th Framework Project. Reference: IST-1999-11253, Start Date: 01-01-00, End Date: 30-06-02, Partners: Alcatel Bell (co-ordinator), Algonet SA, France Telecom R&D, Thales Research Ltd., Inter-University Microelectronics Centre (IMEC), National Technical University of Athens (NTUA), Trans-European Research and Education Networking Association (TERENA), University College London (UCL), University of Surrey) Official Website: http://www.ist-tequila.org

Compared to the TMN architecture in figure 2.3, the individual components are not as strictly divided into layers. However, most components can be assigned to the TMN model. For example, MESCAL features a more sophisticated admission control that works at service subscription level in the management plane, down to the control plane. In TMN terminology the control plane functionality is likely to be part of the Network Element Layer, whereas service subscription spans both Service and Network Management Layers. The MESCAL functional architecture also has functional blocks for managing inter-domain capabilities, such as subscription and invocation of inter-domain links as well as the configuration of BGP policies.

Similarly to TMN, the lower layers take fast decisions on local changes such as router queues, whereas the higher layers take longer term, network wide decisions. Positioned in the highest layer are service and network planning functionality. At this layer human intervention is likely to be found. Beneath this are the core automated management components: traffic forecast, offline traffic engineering and service subscription. These are executed periodically in a process called Resource Provisioning Cycle (RPC). An RPC is triggered when the traffic demands can no longer be accommodated with the current network configuration or more generally when demands and configuration become too dissimilar, crossing a defined threshold value. This process is executed offline on time scales of hours, days or larger. If an RPC cannot provide a suitable configuration, the network provisioning components are alerted.

As before, within the periods of the resource provisioning cycle, dynamic traffic engineering components provide shorter term local decisions to better accommodate the traffic patterns. These dynamic decisions typically take place inside routers and are limited in their decision making scope by boundaries configured through offline traffic engineering, to ensure that the local view of the dynamic component does not distort the global network optimisation. The author has contributed to numerous AGAVE and

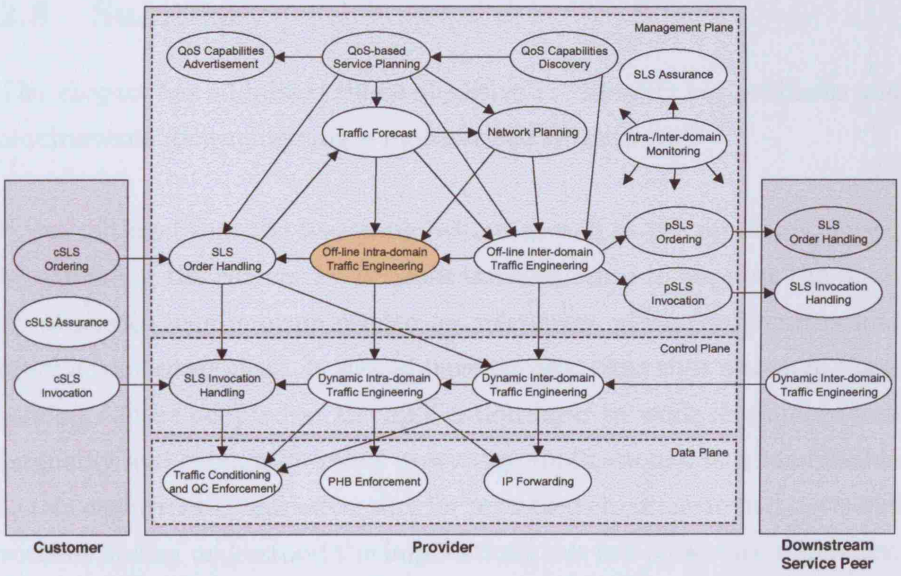


Figure 2.4: MESCAL Functional Architecture [3]

MESCAL deliverables and the focus of this thesis has been in line with a MESCAL project requirement. Because of this relationship, some overlap between this work and the MESCAL project architecture exists throughout this work. A more detailed account of the authors relationship with the projects can be found in appendix A on page 162.

2.8 Summary

This chapter has addressed thesis objective 1: "Identify key problems and requirements for engineering a multiservice IP network."

It was outlined that the reasoning behind as well as the problems caused by extending the current IP network infrastructure to support the quality of service requirements needed for telephony, video conferencing and other advanced services. It should have become clear that although these services can be adopted at the application layer to work reasonably well (arguably well enough for many every-day applications), it is inadvisable to rely on current IP infrastructure for replacing circuit switched networks without having understood the implications this has on service reliability.

Hence the key requirement of service differentiation was identified and it was discussed why service differentiation is important to ensure the coexistence of several services with different QoS characteristics on the same network infrastructure. A definition of QoS was developed and the importance of QoS in IP networks, which divides opinions in the networking community was reiterated.

Several enabling components were identified, which are required in order to turn an IP network into a multiservice network with QoS support. Techniques for engineering the network include network planning and traffic engineering, which are responsible for maximising the capabilities of the network and configuring it in such a way that traffic passing through the network experiences the least possible congestion. Admission control is added to limit the maximum amount of traffic that can enter the network, in order to avoid degradation of already running services. Finally, network management techniques were discussed, which act as the binding element for all components. Network management introduces an oversight where information on topology, traffic monitoring, forecasting and customer subscriptions can be kept together. The network manage-

ment system forms the heart of a multiservice IP network that manages all engineering components and finally ensures that the QoS for each service is guaranteed.

Special focus was placed on the traffic engineering element that will be central to the next chapter. Existing QoS routing algorithms were identified and as well as algorithms for the optimisation of OSPF link weights. Modifying IP and IP routing protocols is difficult, because of the momentum created by the broad existing deployment of infrastructure. Tactics for avoiding the problem are found in mechanisms such as MPLS, but these techniques have their own problems especially in large scale deployment, because of scalability concerns and are therefore widely seen as intermediate solutions. It was pointed out that despite the existing algorithms for control plane QoS routing, the key to IP QoS support is more likely to be to be found in higher layers, management planes configuring a relatively "dumb" network backed up with admission control schemes to control the traffic amounts.

For this reason the next chapter will explore a link weight optimisation scheme that is suited for multiservice networks. The technique builds on the work by Fortz and Thorup [6] and introduces multiple routing planes as well as constraint based routing for service differentiation between the planes.

Chapter 3

Link Weight Optimisation

3.1 Introduction

This chapter and the following, give a detailed description of the IPTE approach that was developed by the author. The split between theory and results was motivated by an attempt to improve clarity of presentation. This chapter contains an in-depth breakdown of the approach and the algorithms that were used. The following chapter presents the simulation results that demonstrate the effectiveness of the approach. In a way, the work described here provides the engine for the traffic engineering management discussion in chapter 6.

3.1.1 Motivation

The preceding discussion of techniques for achieving service differentiation on IP networks has pointed out the complexity of the problems faced by network operators today. IP was not designed to be the foundation for a multiservice, QoS enabled network. It is not designed specifically for a real time critical service like the Public Switched Telephony Network (PSTN). And although the discussion in the previous chapter has focused on the multitude of techniques aiming to enable IP to become a multiservice capable network, the simplicity of IP network design and the apparent ease with which it can be installed are still being used

as selling points. Clearly this only holds true for small and controllable networks, not the envisaged large multiservice networks.

The motivation for building service differentiation on top of distributed routing protocols like OSPF is thus motivated by the hope of being able to show that "traditional" IP techniques can be used to build future multiservice networks and that they do not have to be replaced by more rigid ATM like structures (such as MPLS) that remove a lot of the attractiveness, which made IP networking popular in the first place. The approach in this chapter and the next is building on the work of Fortz and Thorup [6] and presents an alternative to the "micro-management" of traffic in MPLS and shows that QoS aware traffic engineering can be achieved by slightly extending existing technology and without resorting to techniques that cause a large overhead in routers (such as the label information required in each router for each LSP).

3.1.2 Objectives

1. The primary objective is to distribute the network traffic in such a way that all given constraints are honoured for as long as the demands do not exceed the bandwidth specified in the traffic demand matrix.

2. In addition to the primary objective there may be secondary objectives, such as load balancing or pre-planning for future demands, so that arising demands can be satisfied without extensive reconfiguration of the link weights. Although these objectives may also be defined as primary, they can also serve as a guide to ensure a well balanced network and force the optimisation algorithm not to produce "strange" solutions that meet the primary objectives, but are otherwise impractical.

Thus, primary objectives define the "must have" properties of the solution, while secondary objectives define the boundary conditions for the solution as a whole.

3.2 Basic Approach

Distributing traffic demands on OSPF networks is a challenge, because of the "indirectness" of the traffic engineering problem: merely finding optimal paths for each traffic demand as for MPLS is not sufficient, since the desired paths have to be implemented with OSPF link weights. While MPLS allows the explicit pinning of a route between any two nodes and effectively switches packets according to any chosen configuration, OSPF relies on individual routing decisions taken at each node. These are based solely on destination IP address and the networks link weight metrics that determine the shortest path towards the destination. Simply mapping an "optimal" demand distribution as calculated for an MPLS network onto an OSPF network is thus not a realistic goal in most practical cases. Instead, the algorithms for finding a set of suitable paths and those for translating these paths into link weights have to go hand in hand, iteratively searching for better link weights to spread the load across the network.

The approach taken for the IPTE system is thus to distribute the demands on the networks by manipulating the shortest paths between nodes, by carefully choosing the OSPF link weights. In order to do this, the network with routers, link and traffic is modelled in software. Since the problem of calculating the link weights for an optimal allocation of traffic is known to be NP-Hard Jttner et al. [112], the method applied is a heuristic as that of Fortz and Thorup [6]. The shortest path routes between those nodes with non-zero traffic flows are calculated and the link load is deducted by adding each individual flow. A cost function determines which links are the least well balanced and a link weight is selected to be modified in order to redistribute the load. Each time a link weight is manipulated, new traffic loads have to be calculated based on the new shortest path. Of course, manipulating one link weight may modify multiple shortest paths, so that the manipulation of a single link

weight can have unplanned consequences. In order to differentiate traffic, IPTE also considers constraints that can be placed on traffic flows.

Once a set of optimised link weights has been computed, the routers can be configured accordingly. The resulting re-convergence of the OSPF routing tables should lead to the planned traffic distribution if the traffic matrix was accurate. Thus as a result of the optimisation approach, explicit awareness to traffic constraints by OSPF is not required, since it is embedded in the link weights. The approach therefore achieves its objective of achieving traffic differentiation, while leaving the traditional routing of IP networks in place.

3.3 Multi Topology Networks

In an OSPF network, all traffic flowing from one ingress point to the same egress point must follow the same route through the network. It follows that unless DiffServ-like differentiated queuing is performed, all traffic is treated in the same way. This poses limitations on the link weight optimisation, especially if routing constraints are imposed for QoS purposes. For instance, if any of the traffic demands on a source destination pair are delay constrained and a shorter route was chosen for this traffic, all other traffic also has to be routed on the same path in order to accommodate the requirements for only one trunk. Hence a premium route may have to be provided to some low cost traffic as a result of inflexible routing schemes. From a network operators perspective, this means that if many source-destination pairs have only one traffic demand with strict constraints, most of the traffic on the network is routed on premium routes and the advantages of differentiated routing are lost (i.e. load balancing in addition to routing constrained traffic). A further complication to this problem becomes apparent when taking into account that all traffic from any ingress to the same egress might merge into an aggregate flow somewhere in the network. If this is the case, routing constraints from all traffic on these ingress nodes to the same egress node have to be taken

into account when modifying the route of such an aggregate flow. Additionally, large aggregate flows are less flexible in general, with limited choice for traffic engineering because of their capacity requirements.

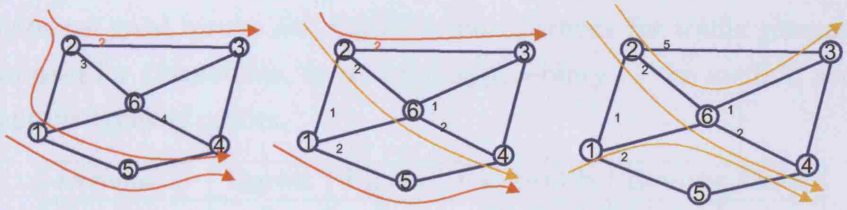


Figure 3.1: Three routing topologies co-existing on one physical network

In order to avoid these problems, routing planes are introduced as an addition to OSPF routing. This could use the mechanisms discussed in section 2.5.4. Both M-ISIS and mt-OSPF are suitable and they are thus used interchangeably in the text. As shown in figure 3.1, each MT-ID marked traffic aggregate is routed individually, based on the corresponding mt-OSPF routing table. While the DSCP is a 6-bit field and theoretically allows 64 routing planes, the MT-ID allows for a theoretical maximum of 128 routing planes, however it is unlikely that such a large number is required. Should mt-OSPF or M-ISIS become accepted by vendors and implemented in router operating software, no modifications would be required in the router software in order to implement an IPTE solution.

3.4 The Demand Matrix

The IPTE optimisation is an "offline" approach that does not dynamically adapt to the traffic patterns in the network. As such, it requires a traffic demand matrix that contains information on the predicted utilisation of the network to be optimised. Additional constraints, such as routing plane membership and QoS constraints also have to be specified if a network operator intends to optimise based on traffic constraints.

Obtaining an accurate traffic demand matrix was discussed in section 2.6. For the purposes of demonstrating the effectiveness of the IPTE approach, no complex method for generating a traffic matrix is required. However, Feldmann et al. [100] states that the TE algorithm used has to be matched with the method for traffic matrix prediction in order to guarantee good results and therefore two methods for traffic generation are used for simulations, to test the applicability of the method across multiple types of matrix.

Demand #	Ingress	Egress	Bandwidth	Routing Plane
0	2	5	0.05	4
1	6	55	0.01	1
		.		
		.		
		.		
8900	33	24	0.04	38

Table 3.1: Format of the Demand Matrix

Table 3.1 gives the format of the demand matrix that is used for the IPTE simulations in the next chapter. For each demand, it defines ingress and egress nodes, as well as bandwidth and routing plane membership. Although the routing plane can be defined if it is a requirement of the traffic demand, it can also be determined by the optimisation algorithm if only load balancing is required. In practise, traffic could be assigned based on a source address hashing in order to avoid splitting of flows, however for the simulations carried out in chapter 5 the demands were randomly split between the classes. While a more clever algorithm for splitting the demands could be conceived, it is left for further study at this point. Two methods for generating demand matrices were used.

1. Flat bandwidth with random ingress, egress and routing plane IDs. These demand matrices provide a simple means of assessing the algorithms performance.

2. Demand generation based on a gravity model. This method was applied by Fortz and Thorup [6] and has been used for comparison

purposes. As in Fortz and Thorup [6] demands were generated according to equation 3.1.

$$\alpha O_u D_v C_{uv} e^{\frac{\lambda_{(u,v)}}{(2\Delta)}} \tag{3.1}$$

Where O and D are random numbers chosen for each node. Similarly, C is chosen for each pair of nodes u, v. The parameter α is a scale parameter, $\lambda_{(u,v)}$ denotes the Euclidean distance between u and v and Δ is the maximum Euclidean distance between any two nodes. This ensures that demands are greater between nodes with shorter Euclidean distance and is a property of gravity models. Also, since there are three random numbers multiplied, the variation between demands is large. For each simulation, a single demand matrix was created. In order to increase the matrices bandwidth requirements, each demand was increased with the scale factor. The demand set for the 10 node topology was created manually; not using the method described above, but rather by applying individual demands across some of the network edges.

3.5 Optimisation Constraints

3.5.1 General

The optimisation of link weights for a single routing plane was shown to be NP-hard by Fortz and Thorup, therefore the algorithm is based on a heuristic iteration seeking to reduce a cost function. Any constraint that can be formulated as a cost can be taken into account when performing a run. The word "optimisation" in this context means to find a link weight configuration with the smallest cost that the algorithm can find in a defined amount of time. The scope for possible constraints is limitless, however, too many constraints will dilute the cost function and reduce the quality of the optimisation. It is also important to be careful when configuring potentially conflicting constraints so that they do not cancel each other out. Constraints of different priority can be configured though

a weighting in the cost function ranging from very high importance (discard the solution, if it does not meet a constraint) to very low (accept solution, but prefer one that matches given constraint).

3.5.2 Load Balancing Constraints

Load balancing is useful for maximising the value of all network components by spreading load more evenly across the whole network. Through the reduction of utilisation peaks on some links, balanced networks can potentially absorb changes in traffic load better than an unbalanced networks. The concept can be taken further with multi topology routing. For instance, load balancing does not have to be performed evenly, it can also be used to prepare for expected changes in link load, such as daily traffic peaks.

3.5.3 Resilience Constraints

Protecting the network from possible link failures or enabling the replacement of a link during a maintenance cycle could be facilitated by ensuring that the next shortest path in the Dijkstra tree has sufficient capacity to cope with the additional load from the failed link. By using multiple route topologies, an additional layer of safety can be constructed through having an "empty" topology configured with a backup configuration that is taking into account one or more particular failed link(s). In the case of failure, the backup topology with ready configured link weights and converged routing tables can be enabled simply by configuring the ingress MT-ID marking to switch the traffic to the new routing topology.

3.5.4 QoS Constraints

In order to provide support for QoS traffic, constraints of delay, jitter and packet loss probability can be imposed through DiffServ PHBs throughout a single routing plane. In addition to PHB prioritising, multiple routing planes provide the option of configuring differentiated routing and select less congested paths or shorter paths than those used for best-effort traffic.

It was argued by Ma and Steenkiste [113] that if a weighted-fair-queuing scheduling algorithm was used, the end-to-end delay and jitter bounds become functions of the reserved bandwidth. Trimintzios et al. [114] use this to formulate QoS constraints as hop count limitations as follows:

- In order to ensure an upper limit on queuing delay, a maximum queue length has to be defined per PHB. The queue length limit is enforced by dropping excess packets and queuing delay can be enforced by imposing a hop count constraint. Since jitter is related to queuing time variance for different packets, this also limited by enforcing a maximum queue length.

- Some rough assumptions can be made about the packet loss probability and the end-to-end delay, if in addition to queue length, the utilisation of the network is known. In this case the packet drop rate becomes a function of queue length and number of hops. End-to-end delay is then the sum of the average queue length and the transmission line delay.

- If assumptions are made on average queue length and a strict maximum queue length is imposed, the correct distribution of link loads and prevention of congested links has to be ensured. This can only be done if the traffic demands do not exceed the ones specified in the demand matrix that was used for the link weight calculation. Some degree of excess link load may be tackled by over-specifying the demands for the link weight calculation. However, an admission control policy is unavoidable for quality of service enforcement, to ensure that the traffic does not exceed the planned capabilities of the network. It is traffic engineering that has to specify how much traffic can be admitted by the admission control policy and thus an incorrectly specified upper limit will result in admission control allowing too much traffic to enter the network.

While the assumptions presented above may help to achieve some QoS differentiation, the argument of reducing loss, jitter and delay to a hop

count constraint seems somewhat weak, especially considering the discussion on queuing in section 2.3.3. It is rather more likely that the combination of several TE schemes will lead to an acceptably high probability of achieving the required quality. These factors are as outlined above: a well balanced load to reduce utilisation spikes, the shortest possible path or that path through the network with the highest capacity links to reduce the probability of congestion and prioritised queuing through DiffServ to ensure the capacity is allocated to the most important traffic. In addition to these techniques, a QoS aware management system is required. This is discussed in chapter 6. The effects of queueing are not modeled in this thesis and the optimisation constraints in this chapter are based on propagation delay as well as hop count.

3.6 Algorithm Description

3.6.1 Outline

This section describes the steps of the link weight computation algorithm. The activity diagram in figure 3.2 illustrates the process. As each item is described in more detail later on in this chapter, more complexity is introduced and the order of the steps is changed. At this point, the emphasis is placed on clarity.

1. Initialise the link metrics by setting some arbitrary (e.g. unit value) weights for each link and each MT routing plane. This defines the initial condition of the weight setting algorithm.

2. Calculate the shortest path tree for each ingress-egress pair that has a non-zero traffic load. This has to be repeated for each MT routing plane.

3. Calculate the total link load for all links and hence identify the highly congested links. Each routing plane has to be taken into account individually, although the total link load is a single figure representing total effective link load across all planes.

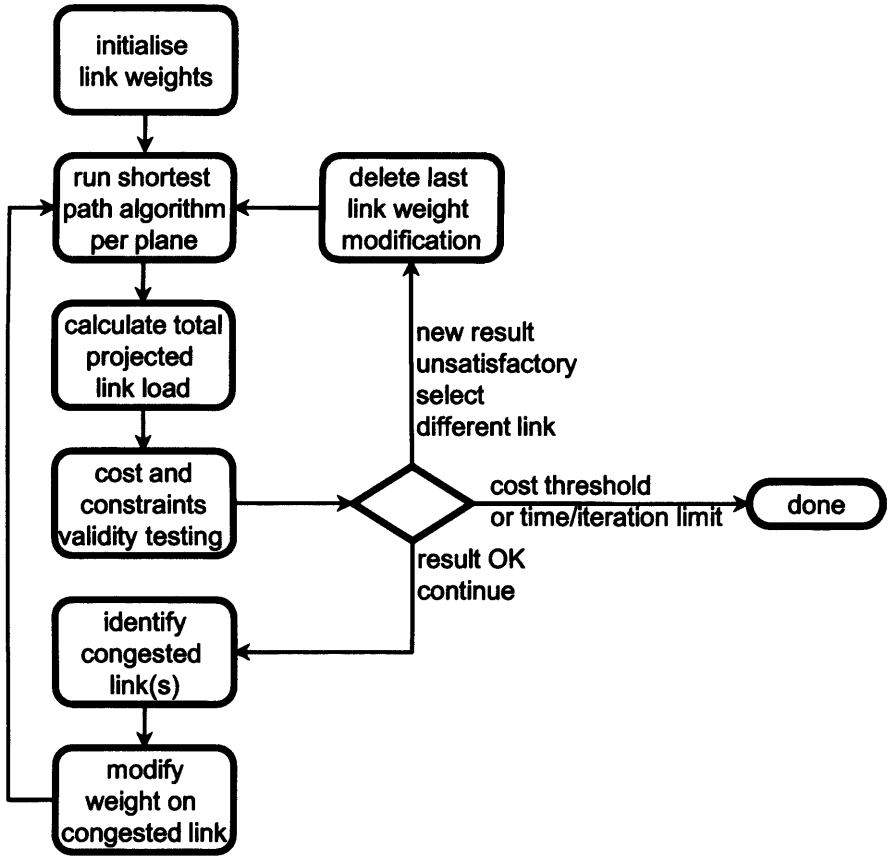


Figure 3.2: The Basic Link Weight Optimisation Algorithm

4. Cost and constraints validity testing. The cost of each link is determined in this step as a function of the link load. If the latest solution performs worse than an earlier one, the current result is deleted. If the result is found to be satisfactory, the algorithm continues.

5. Identify congested links. This is the selection of a link and a routing plane on which a link weight modification will be performed. The selection depends both on the cost determined in the previous step, as well as previously tried configurations.

6. Modify weight on congested link. Increase the link weight on the selected link and MT plane.

Steps 2 to 6 are iterated until the decision point during step 4 determines to stop the algorithm. The decision to stop can be defined as a fixed number of iterations, time or a cost function, based on a weighting of factors, such as how long there has been no improvement in cost.

3.6.2 Routing Model Definitions

In order to assemble the weight manipulation algorithm, it is beneficial to define some of the reoccurring terms. As in Awduche et al. [115] the network is modelled as a directed graph, $G = (N, E)$ where the nodes $n \in N$ and links $l \in E$ represent routers and links between routers. A link l has capacity $c(l)$ indicating the amount of traffic that l can accommodate. The traffic is given in form of a demand matrix D, representing the amount of traffic flowing on a path between any nodes (s, d). This demand matrix is provided by traffic forecast, which in turn is calculated based on historic data and intra-domain traffic demands. It can be expected that many of the demands (s, d) are zero, as not every ingress/egress pair has traffic flowing between it. The routing problem is to distribute the traffic from non zero $D(s, d)$ across the network evenly. The load on a link is given as x_l, this is the sum of all demands $D(s, d)$ using the link l. The utilisation of l is then given by $x_l/c(l)$. For the weight allocation, a weight ω is assigned to each MT-ID $h \in H_l$ on each l, so that $\omega_{h,l}$ denotes a unique weight.

3.6.3 Initialising Link Metrics

This step defines the initial conditions for the weight setting algorithm. Depending on the initial conditions, the algorithm can lead to faster convergence as well as better results. The most obvious choices are unit, inverse capacity or random weight settings. In operational networks, inverse capacity weights are often used based on a recommendation by Cisco Systems Halabi [116]. The reason for this is that using inverse capacity link weights, OSPF will automatically favour the larger links towards the

core of the network. Inverse capacity link weights are thus a crude, but practical approach to traffic engineering.

It has been shown in Fortz et al. [117] that initial weight settings based on inverse link capacity, greatly improve the convergence time of the algorithm. This seems sensible, since when compared to a random weight assignment the solution with inverse capacity weights is a better traffic engineering solution and hence a better starting point for the link weight optimisation algorithm. However, early simulation results from this thesis have shown that although convergence is achieved more rapidly, the inverse capacity link weights put a large emphasis on a particular solution and may force the algorithm to converge on a local minimum in the search space.

3.6.4 Calculating the Shortest Path Trees

This needs to be done for each load bearing ingress egress pair. However, because each MT routing plane has a unique set of link weights, the shortest path tree has to be calculated for each ingress egress pair and for each plane. It is essential that the algorithm for calculating the path is identical to that of the routers inside the network, otherwise the routes in the network might be different than those simulated during the optimisation, causing congestion or other unforeseeable effects.

3.6.5 Calculating Total Projected Link Load and Cost

Summing Individual Link Contributions

The total link load is calculated by distributing the traffic from each ingress egress pair over the shortest paths calculated in the previous step. There is no explicit partitioning of bandwidth between planes on a link, thus link loads from each routing plane are summed used to find the total free capacity of each link. Bandwidth on each MT plane can be associated with an *equivalent* bandwidth factor. This factor can be used to accommodate for traffic with larger bandwidth variance, by factoring

a larger margin or for higher priority traffic. For each MT h of a set H_l of MTs on a link with bandwidth allocation $x_{l,h}$, the equivalent bandwidth can be expressed as a function $f_{l,h}(x_{l,h})$ increasing in $x_{l,h}$ and greater for any given $x_{l,h}$ with higher priority h. The total equivalent load of each link is then given by (3.2).

$$\sum_{h \in H_l} f_{l,h}(x_{l,h}) \tag{3.2}$$

Assuming that $c(l)$ is not the same for all links, L_e is the normalised utilisation of the link.

$$L_e = \frac{\sum_{h \in H_l} f_{l,h}(x_{l,h})}{c(l)} \tag{3.3}$$

In order to arrive at an overall cost function Φ, the statement has to be extended to reflect a cost per link which can then be summed over $l \in E$ to form (3.4).

$$\Phi = \sum_{l \in E} \Phi_l(L_e) = \sum_{l \in E} \Phi_l \left(\frac{\sum_{h \in H_l} f_{l,h}(x_{l,h})}{c(l)} \right) \tag{3.4}$$

Exponential Cost Function

The cost function is based on the observation that high link utilisation causes unwanted effects, such as delay and packet loss. Even solutions that have higher average utilisation on several links are better than solutions with one over-utilised link. Therefore in order to capture this, when comparing different solutions, those with unnecessary highly utilised links need to be highlighted. This done by assigning a disproportionately high cost links with high utilisation. In this case, a curve similar to the one used by Fortz and Thorup [6] is used, which has an exponentially increasing curve. Other functions such as x^3 or that of the M/M/1 queueing delay $\frac{1}{\mu - \lambda}$ (equation 2.1) would be sufficient, as long as they exhibit a steep increase in cost at higher utilisation. As in Fortz and Thorup [6],

discrete values are used in order to reduce calculation time of individual cost values for each link. The actual curve used for simulation is described by equation (3.5).

$$\phi_l(L_e)' = \begin{cases} 1 & \text{for} \quad 0 \leq L_e \leq 1/3 \\ 50 & \text{for} \quad 1/3 \leq L_e \leq 1/2 \\ 100 & \text{for} \quad 1/2 \leq L_e \leq 7/10 \\ 500 & \text{for} \quad 7/10 \leq L_e \leq 9/10 \\ 1500 & \text{for} \quad 9/01 \leq L_e \leq 1 \\ 4000 & \text{for} \quad 1 \leq L_e \leq 11/10 \\ 5000 & \text{for} \quad 11/10 \leq L_e \leq 3/2 \\ 10000 & \text{for} \quad 3/2 \leq L_e \leq \infty \end{cases} \tag{3.5}$$

Increasing the cost beyond 100% utilisation is a necessity introduced by a special case of the heuristic algorithm, which sees lower cost when further loading already overloaded links in order to reduce cost on other links in the network. Therefore, while the exponential cost increase should deter the algorithm from creating these relatively low cost solutions that contain a few extremely congested links, the cost curve is additionally defined to increase beyond 100% link utilisation to add futher disincentive from creating such solutions.

3.6.6 Cost and Constraints Validity Testing

Test if cost has increased or decreased since the last iteration. The algorithm does not allow non-improving moves, so if the cost has increased the previous weight modification is discarded.

3.6.7 Reducing Load on Congested Links

In order to minimise the cost function, those links with the largest contribution to the total cost have to be identified.

$$l_c = \max_{l \in E} \Phi_l(L_e) \tag{3.6}$$

Equation (3.6) identifies such a link. Before modifying any weights, it is necessary to choose a candidate traffic flow passing though l_c which to modify. There are several MT routing planes to select from. The algorithm first selects the routing plane with the largest proportion of flow, then the second and finally the third largest before progressing to the next high cost link. It may be a good option to avoid modifying traffic mapped to high priority MT planes and modify first the lower priority traffic. This would ensure that high priority traffic stays mapped to large capacity links and it also guarantees least disruption of this traffic in case of re-engineering of the network, such as discussed in more detail in chapter 6. The high priority traffic will not be as affected if its paths do not change. This has not been validated in simulation, since the transient

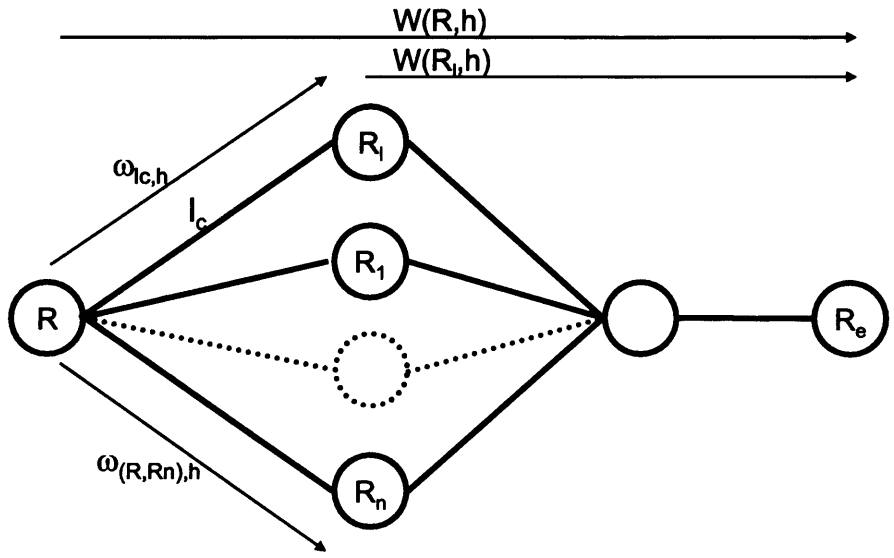


Figure 3.3: Modifying the Shortest Path

effects of changing link weights only become apparent in a packet level simulation or a test bed configuration.

As illustrated in figure 3.3, let R be the origin node of link l_c, and let R_l be the destination node of l_c. Let $W(R,h)$ be the sum of the weights on the shortest path from R to some egress node R_e on routing plane h.

Also let R_n be a neighbouring node to R of a set of neighbours $R_n \in B$. Equation (3.7) shows the length of different paths leading from R to R_e.

$$W(R,h) = W(R_l,h) + \omega_{(l_c,h)} \leq W(R_n,h) + \omega_{(R,R_n),h} \qquad \text{for all } R \in B \tag{3.7}$$

The objective now is the reduction of the load on l_c, by redistributing some of its load onto other neighbouring links.

Modifying a single link weight

For this single weight modification, the neighbourhood of node R is searched in order to locate a neighbouring node R_n such that

$$W(R_n,h)+\omega_{(R,R_n),h} < W(R_l,h)+\omega_{l_c,h}+\varphi \text{ where } \varphi \text{ is a weight adjustment} \tag{3.8}$$

It is desirable to choose a small value of φ. The reason is that the adjustment of a weight may cause other routes to change. By choosing the smallest weight change, the probability of such unwanted changes is kept low. At each iteration the algorithm creates a list of links ordered by their congestion level and then moves though the list using the following method for incrementing weights:

1. Choose the most congested link and choose the largest contributing routing plane, increment weight by 1.
2. Try second and third largest contributing plane and increment weight by 1.
3. Choose a random plane and increment link weight by number between 1-5.
4. Repeat steps 1 to 3 for next link in the list. If an change is successful, recompute list.

This stage of the algorithm does not allow for non-improving moves, so there are no stability issues to be considered. Non-improving moves are

introduced with the perturbation feature of the algorithm that is discussed in section 3.6.10.

Using ECMP

Large aggregated flows develop naturally in networks with shortest path algorithms, as a result of traffic aggregation from multiple ingress points towards one single egress point. If these different traffic flows meet on a node, they become a single aggregate flow. A network consisting of such large aggregates is harder to optimise than one consisting of many small flows. As discussed previously, ECMP can be used to split the traffic when more than one shortest path route is available. By setting the $W(R,h)$ equal to some or all of $W(R,h) \in B$ so that all those R_n lie on the shortest path. In order to do this (3.9), should all be set equal.

$$W(R_l,h)+\omega_{(R,R_1),h} = W(R_2,h)+\omega_{(R,R_2),h} = \cdots = W(R_n,h)+\omega_{(R,R_n),h} \tag{3.9}$$

It is important that (3.10) holds true for this modification, as a reduction in weights may cause loading of a link that is already heavily loaded.

$$W(R_n,h)+\omega_{(R,R_n),h} \leq W(R_y,h)+\omega_{(R,R_y),h} \quad \text{where } R_n \in B, R_y \notin B \tag{3.10}$$

The drawback of this approach is that a carefully set up multi path can be damaged when further iterations of the algorithm modify paths nearer the sink of the ECMP traffic. These modifications could cause some of the multiple shortest paths to become longer or shorter than others and thus attracting or repelling all of the traffic. Care has to be taken in order to prevent these scenarios from occurring by e.g. applying the ECMP rules only in the last iterations, or as a last resort. It is also possible to conceive of keeping a database with all the ECMP weight changes in order to keep a record of which links have been modified. The ECMP method is not validated though simulation, as it seemed too likely that later modifications in link weight would destroy the fragile balance. More importantly however, since the advantages of ECMP are similar to those

of using multiple routing planes, the impact of additionally using ECMP is likely to be limited.

3.6.8 The Cost of Constraints

Hop Count and Link Propagation Delay

The hop count and propagation delay constraints can be based on known values of link propagation delay or simply on router hop count and can be used to avoid satellite links as well as unnecessarily long paths through the network for delay sensitive traffic. For the purposes of simulation the main argument is that this constraint is an end-to-end additive cost as opposed to a per-link additive cost. This means that the cost incurred by each demand can only be determined by inspecting all the links it uses at the same time and adding each links respective delay. Since the optimisation algorithm as discussed so far is based on identification of high cost links, in order to factor delay constraints into the cost function it is necessary to calculate the end-to-end path of each demand on a link. Once the total link delay has been determined, a defined per-link cost d_h[1] is added and the total hop count cost can be calculated by summing the delay introduced costs on each routing plane[2] (3.11),

$$\sum_{h \in H_l} d_h \tag{3.11}$$

which is then added to the equivalent utilisation of the link as the sum of d_h over all (delay constrained) routing planes (3.12).

$$\Phi = \sum_{l \in E} \Phi_l(L_e) = \sum_{l \in E} \Phi_l \left(\frac{\sum_{h \in H_l} f_{l,h}(x_{l,h})}{c(l)} + \sum_{h \in H_l} d_h \right) \tag{3.12}$$

[1]While d_h is assumed to be a constant here, it can be defined as a cost function itself, based on link delay and different between classes.

[2]While the sum is performed over all routing planes, it is expected that not all routing planes have delay limitations and even if they do, not all are exceeded at the same time.

When deploying link delay constraints it is best to define the initial link weights proportional to delay for the routing plane in question. This ensures that the lowest link delay path is already configured for each demand. The cost function ensures that the algorithm balances the requirements of all routing planes and refrains from lengthening the paths on a delay constrained plane for the purposes of load balancing (or other), unless this causes greater cost than lengthening the path. The cost function of each constraint ensures that it receives the desired priority.

The actual values for d_h have to be calibrated for each network topology, since they depend on the maximum hop count. If d_h is configured for a network smaller than the one that is being optimised, penalties will be given to paths that are already short. Equally, if the penalties are configured for a network larger than that which is optimised, no penalties will be given when they should be. Penalties range from 0 to 1 and add to each links utilisation. Thus, a potentially large penalty is incurred for a path with many hops and it will be discouraged.

An alternative method for configuring a routing plane with short paths is to apply unit weights to this plane and exclude it from optimisation. This approach works if the plane in question does not contain a large proportion of the total traffic and the optimisation can rebalance the network by load balancing the remaining network planes.

Available Bandwidth

Apart from low delay paths though the network, the second type of constraint discussed here maximises available bandwidth on a routing plane. This is useful to reduce the packet loss, jitter and queuing delay as well as increase robustness for traffic variability. In order to increase the free capacity of paths belonging to one class, the equivalent bandwidth of the class is increased. This spare bandwidth factor β multiplies the real bandwidth of the class, hence causing higher cost for traffic on this class. As a result, the IPTE process allocates traffic of such a class more spare

capacity. The new term extends the equivalent bandwidth equation (3.2) in the cost function.

$$bw = \sum_{h \in H_l} f_{l,h}(x_{l,h}) * \beta_h \tag{3.13}$$

Since the approach allocates spare capacity only for demands specified in the demand matrix, the spare capacity does not apply throughout the plane, but exclusively to the predefined traffic trunks. This has the advantage of not wasting capacity where it is not required. As there is no partitioning, all traffic on planes sharing these links will also experience the lower utilisation, however, they are unlikely to share all of the links.

The range of possible values of β_h is subject to the size of the network and the significance of the fraction of bandwidth it acts upon. If a large amount of traffic is affected by β_h, the available bandwidth will soon be consumed. Because of this, the approach lends itself well to replacing MPLS tunnels, multiple of which could be defined through one or two routing planes.

Further Constraints

The constraints illustrated in this section are a small subset of the possible constraints that could be implemented through the cost function. As outlined in section 3.6.8 further possibilities include link failure backups, which would require some routing planes to avoid particular links or nodes. Also, combinations of several constraints can be formulated for different classes.

3.6.9 Undesirable Consequences of Reducing Cost

Although modifications made to link weights should, in general, succeed at redistributing the traffic flow in question, they may have undesirable side effects. A full view of changes that a link weight modification has

caused, becomes visible after the shortest path algorithm has been executed with the new set of weights and costs have been calculated for each link. Two problems may occur:

- Traffic shifted away from one link has caused the exceeding of constraints on a neighbouring link.
- The route modification has caused a constraint for the rerouted traffic to be exceeded. This could be the case for the traffic that was chosen for re-routing and also for other traffic that was rerouted through the weight change.

If neither are true, the weight modification can be accepted and the algorithm continues. If a problem occurs, the weight modifications have to be discarded (and also banned from being chosen again) and other, different modifications have to be identified to redistribute the traffic.

3.6.10 Perturbations

The purpose of perturbations is to avoid for the algorithm to become trapped in a local minimum of the search space. This is especially likely to happen in cases with more routing planes, since single weight changes have an increasingly smaller impact with increasing number of planes. Perturbations disturb a percentage of the link weights across all routing planes to nude the solution out of local minima. Once a perturbation has begun, a higher cost than the previous best cost is tolerated for a number of iterations (a value of 200 iterations was found to work well for topologies with up to 50 nodes) to let the perturbed solution converge before comparing it to the previous solution. If the new solution is found to be better, it is kept, if its is worse than the previous solution is its discarded and a new perturbation is initiated. No stability concerns should arise through perturbations, since non-improving moves are only tolerated for the duration of the perturbation. Once the perturbation has run for the defined number of iterations, it is only kept if it has improved on the original starting weight configuration.

3.6.11 Full Algorithm Description

The diagram in figure 3.4 shows the complete algorithm. It is based on the basic steps as outlined in figure 3.2, but now additionally takes into account the more detailed discussion from the previous sections. The

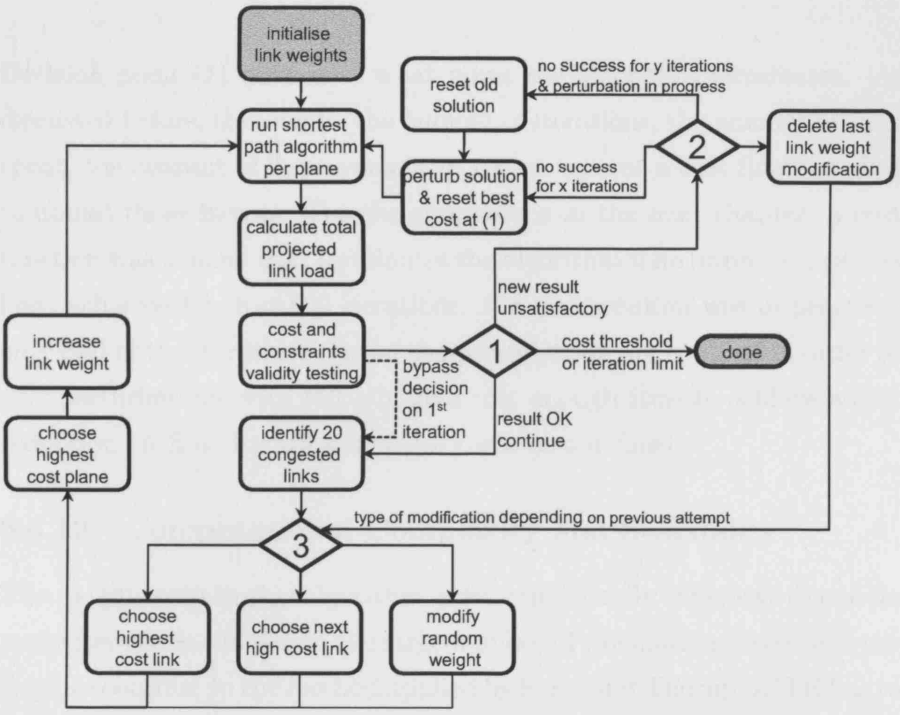


Figure 3.4: Full Algorithm

selection of a high cost link is determined at decision point (3). First the highest cost link as well as the largest contributing routing plane is chosen, if this fails, the second most cost contributing routing plane on the same link is also tried. If this also fails, the algorithm progresses to try the next high cost link in its list until either all links have been tried or a perturbation is initiated at decision point (2). In order for the heuristics

not to become stuck on a local minimum, the solution can be perturbed if no progress is made for a number of iterations x. For a perturbation, 10% of link weights on all active routing planes are randomly chosen

and incremented or decremented by 1. The resulting configuration is optimised for a number of iterations y in order to see if it can perform better than the previous best solution. If no improvement is noticed, the perturbed solution is discarded and a new perturbation is initiated. If a perturbed solution becomes successful it is retained.

Decision point (1) defines at what point the algorithm terminates. As discussed before, this can be the number of iterations, the amount of time spent, the amount of improvement in recent time or a cost function that combines these factors. For the simulations in the next chapter, a cost function was defined that terminates the algorithm if no improvement has been achieved for over 500 iterations. If a perturbation was in progress, only half of the iterations during the perturbation are counted in order to give perturbations with initially high cost enough time to achieve a cost reduction. A final iteration boundary was also defined.

3.6.12 Computational Complexity and Scalability

The weight optimisation algorithm is computationally expensive due to its many iterations and due to the large number of calculations necessary per step. In contrast to the method applied by Fortz and Thorup, IPTE has to perform additional Dijkstra and link capacity calculations, depending on the number of deployed routing planes. For each additional routing plane, calculations of routing tables and per link load calculations have to be made individually, to factor into the total load and cost calculation. While the number of routing planes is theoretically limited to the particular mechanism used (64 if the DSCP field is used, 128 if the MT-ID field is used and potentially many more if an optional field in the IPv6 header were used), it is unlikely that so many planes are required.

With a large number of planes, IPTE also requires a larger number of traffic demand pairs to perform load balancing effectively. If the number of demands is too low, each routing plane might only carry few demands over any link. This reduces the effect of a link weight modification and

thus increases the amount of "optimisation work" required by the IPTE simulator. Until Wang et al. [8] published their k-set optimisation work, this was the only work based on link weight optimisation to assume weight settings for multiple routing planes and much of the computational requirements have therefore been unclear. However, ongoing advances in processor design counteract this argument.

3.7 Differences to Fortz and Thorup

While the algorithm presented here is based on the work of Fortz and Thorup [6], it differentiates itself from the original algorithm in two main points:

1. Multi Topology Optimisation - The introduction of multiple virtual topologies on one physical topology enables multiple routing strategies. The parallel operation of a number of routing planes solves a major shortcoming of SPF based routing protocols, in allowing traffic with the same source and destination addresses to be routed on different paths. While the most obvious application of this new capability is to improve load balancing on the network, it also makes multi-constraint routing possible. Since explicit bandwidth partitioning is not possible in OSPF networks, the algorithm of Fortz and Thorup was extended in this respect to calculate a common cost value over all active routing planes. In addition to this, the rules for selecting a candidate link weight were modified to choose between different routing planes.

2. Multi Constraint Optimisation - The original algorithm by Fortz and Thorup was intended for load balancing the network. This algorithm introduces constraints other than load balancing to enable traffic engineering for QoS routing purposes. While multi-constraint optimisation on a single routing plane is likely to lead to conflicts between individual goals (for instance the shortest path for delay

sensitive traffic may not be the widest path for bandwidth intensive traffic), the multi topology approach of this algorithm enables multiple constraints to be implemented on individual routing planes. This makes it possible to optimise one plane for delay sensitive traffic, while another can be optimised for bandwidth sensitive traffic. Further planes can be used to load balance the network with best effort traffic.

3.8 Summary

The goal of the proposed traffic engineering approach is to compute a set of OSPF link weights, which optimise the networks routing towards given optimisation goals. These goals can include, load balancing, improved resilience to failures or traffic QoS constraints. Several goals can be combined to form optimisation policies. The proposed solution is built on classical OSPF routing, but additionally introduces multi topology awareness to implement multiple routing planes. Each routing plane is assigned its own link metrics and can thus route traffic independently.

Given a traffic demand matrix and the network topology, the algorithm computes a set of link weights using a search heuristic. The optimisation is cost function based and the optimisation policies are taken into account by factoring each individual goal into the cost function. Being based on IP routing, the TE solution is more lightweight than MPLS-TE in terms of state-information required to be maintained by the network and the associated management overhead for network configuration.

The algorithm presents an evolution of the algorithm formulated by Fortz and Thorup. The emphasis of IPTE lies with providing proof of concept for using link weight optimisation to ends other than single plane load balancing, this algorithm does not lay claims on speed or efficiency. Strictly speaking, this algorithm could be classed as genetic with one parent (i.e. mutations only), however, it is difficult to compare it to any of the algo-

rithms in the literature unless these algorithms were extended to support multiple routing planes and constraints.

Since QoS information remains in off-line algorithms, no QoS awareness is required at layer 3. Recent mt-OSPF Internet Drafts provide the required MT routing support, so potentially no major changes at the router level are required for the approach to be realised.

Chapter 4

Simulator and Functional Validation

4.1 Introduction

This chapter describes the optimisation software that was built in order to test the algorithms from the previous chapter and of the plane transitioning algorithm in the final chapter. The chapter contains three elements: design and operation of the software, discussion of computational efficiency and a short introduction to additional software tools that were built for data collection.

4.2 Software Tool

4.2.1 Justification

In order to analyse the performance of the IPTE algorithms it was necessary to implement the iterative weight setting algorithm in software. None of the standard network simulators are suitable as starting point, since most are designed for packet or flow level network analysis. None can be modified easily to analyse link weight metrics. It was therefore decided to build a new implementation, tailored for iterative link weight analysis. More recently, the TOTEM toolbox has become available [7],

however by this time, the implementation was already complete. The implementation is based on parts of the graph data structures that are used by the "BRITE" topology generator developed by Medina et al. [118], since this contained some useful hash map based data structures.

4.2.2 Design

Overview

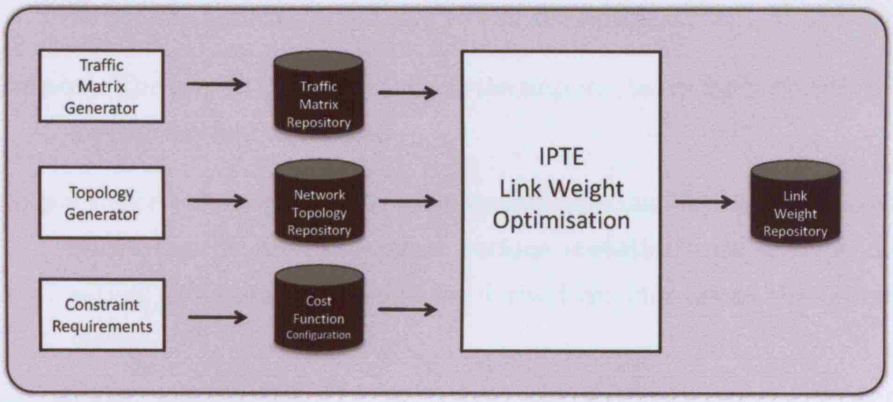


Figure 4.1: Software Input and Output

The main loop of the simulator iterates on the algorithm described in section 3.6.11, with its inputs and outputs shown in figure 4.1. Additionally, when required, the simulator can also take added constraints from section 3.6.8 into account. The output to the application is a set of link weights that could potentially be implemented in either test bed or flow/packet level simulator. The software platform is Java based and implements the same Dijkstra shortest path algorithm, also used in routers. As shown in figure 4.1, in addition to internal settings the software requires the following input data in order to compute a set of link weights:

- intra-domain network topology such as BRITE and Rocketfuel inferred topologies
- a traffic matrix
- any QoS constraints and additional optimisation policies

Output generated by the software platform is stored in a link weight repository.

Software Structure

The basic structure of the IPTE software is the similar to that the BRITE topology generator. It contains of 6 packages:

Graph The Graph package contains the core java classes used to construct the network graph and to run the optimisation.

Import The import package contains the import classes for both BRITE and Rocketfuel topologies.

Export The export package contains some experimental export classes, which can be used to extract various statistics from the optimisation. The main statistics are derived directly inside the Graph class.

TrafficGen This package contains the traffic demand generator.

Util Package containing additional utilities such as the error reporting tool. Also contains the average class which can be used for output data analysis.

simulator This package contains the main function and wrapper classes for the simulator, the traffic matrix generator and the average class. Most settings can be configured with the wrapper classes.

4.2.3 Configuration and Operation

The Job File

The job file is used to specify individual optimisation jobs to be carried out by the simulator. For each job, input and output files are specified along with several other options. Each job is specified in one line with the following space separated parameters:

Topology Path - String Specifies the path to the topology file including the name and prefix of the topology file. Topologies can be specified either in BRITE format or in Rocketfuel format.

Demand Path - String Specifies the path to the demand files. Can address multiple files at the specified location. Demand files have to take the format described in section 3.4. Although both bandwidth and routing plane membership need to be specified, these values can be adjusted or overridden by the two parameters that specify bandwidth scaling (directly below) and the parameter that specifies the number of routing plane optimisations (further below).

Number of Bandwidth Scalings to be Performed - int If this is grater than 1, several optimisations are performed, scaling the demands with by a factor increasing by one until the specified number is reached. The demand files generated are stored alongside the original in case they are required in a later optimisation.

Initial Bandwidth Scale Factor - double Factor used to scale all demands as specified. Can be used to adjust the average bandwidth consumption of a demand set.

Results Path - String Specifies the path to the results.

Link Weights Initial Condition - String Can take the values of "unit" for unit link weights, "inversecap" for inverse capacity link weights and "random" for random link weights.

Topology Type - String Can take the value of "brite" for a topology in the BRITE format and "rocket" for a topology in Rocketfuel format.

Number of Routing Plane Optimisations to be Performed - int If this is greater than 1, several optimisations are performed, with demands randomly assigned to 1, 2, 4 and up to 8 routing planes. The demand files generated are stored alongside the original in case they are required in a later optimisation.

Other Configuration Options

Several other configuration options are available, such as the specification of bandwidth and hop count constraints and triggers for beginning and ending the optimisation cycle and perturbations. These have to be made in the program code itself at this point although external configuration handles could be implemented.

Command Line Execution

The command line allows for the specification of the job file and takes the form:

```
java -Xms32m -Xmx800m -jar iptesim.jar <JobFile>
```

The parameters **-Xms32m** and **-Xmx800m** are java specific and indicate a minimum and maximum memory bound for the virtual machine. A maximum value of 800 MB is recommended for topologies up to 100 links. The parameter **-jar** specifies that the java program to be executed is a jar archive. **iptesim.jar** is the optimisation program itself and **JobFile** specifies a job file.

The text below shows the typical console output from the optimisation software, the output shows the beginning and the end of the cycle with a large part of of optimisation removed from the middle as indicated.

```
[MESSAGE]: Parsing BRITE format file
[MESSAGE]: Weight Init Mode: unit
[MESSAGE]: Number of nodes: 10 Number of Edges: 34 Number of Demands: 500
[MESSAGE]: Cost: 6.857200000000002 Cost/Uncap: 0.8809352517985658
[MESSAGE]: Uncapacitated: 7.783999999999962
[MESSAGE]: Parsing BRITE format file
[MESSAGE]: Weight Init Mode: unit
[MESSAGE]: Class0 hop count min: 0.0 max: 5.0 avg: 2.22 stdev: 1.16 num: 224 Util: 10.55 UtilStdev: 10.53
[MESSAGE]: Class1 hop count min: 0.0 max: 5.0 avg: 2.23 stdev: 1.19 num: 276 Util: 10.55 UtilStdev: 10.53
[MESSAGE]: Class2 hop count min: 0.0 max: 0.0 avg: ? stdev: -0 num: 0 Util: ? UtilStdev: -0
[MESSAGE]: Class3 hop count min: 0.0 max: 0.0 avg: ? stdev: -0 num: 0 Util: ? UtilStdev: -0
[MESSAGE]: Class4 hop count min: 0.0 max: 0.0 avg: ? stdev: -0 num: 0 Util: ? UtilStdev: -0
[MESSAGE]: Class5 hop count min: 0.0 max: 5.0 avg: 2.22 stdev: 1.18 num: 500 Util: 10.08 UtilStdev: 14.33
Mean Delay Class 0: 2.22 Class 1: 2.23 Class 2: ? Class 3: ? Other: ?
Mean Util  Class 0: 10.55 Class 1: 10.55 Class 2: ? Class 3: ? Other: ?
[MESSAGE]: Parsing BRITE format file
[MESSAGE]: Weight Init Mode: inversecap
[MESSAGE]: Class0 hop count min: 0.0 max: 5.0 avg: 2.54 stdev: 1.46 num: 224 Util: 9.77 UtilStdev: 9.75
[MESSAGE]: Class1 hop count min: 0.0 max: 5.0 avg: 2.39 stdev: 1.35 num: 276 Util: 9.77 UtilStdev: 9.75
```

```
[MESSAGE]: Class2 hop count min: 0.0 max: 0.0 avg: ? stdev: -0 num: 0 Util: ? UtilStdev: -0
[MESSAGE]: Class3 hop count min: 0.0 max: 0.0 avg: ? stdev: -0 num: 0 Util: ? UtilStdev: -0
[MESSAGE]: Class4 hop count min: 0.0 max: 0.0 avg: ? stdev: -0 num: 0 Util: ? UtilStdev: -0
[MESSAGE]: Class5 hop count min: 0.0 max: 5.0 avg: 2.46 stdev: 1.4 num: 500 Util: 9.34 UtilStdev: 13.28
Mean Delay Class 0: 2.54 Class 1: 2.39 Class 2: ? Class 3: ? Other: ?
Mean Util  Class 0: 9.77 Class 1: 9.77 Class 2: ? Class 3: ? Other: ?
[MESSAGE]: Progress: 0.0% Current Best: 6.857200000000002
[MESSAGE]: Changed weight on routing plane: first plane:1 on link: 131073 Cost: 6.668200000000005

<<truncated>>

[MESSAGE]: Beginning perturbation...
[MESSAGE]: Changed weight on routing plane: first plane:0 on link: 458758 Cost: 6.129900000000002
[MESSAGE]: Changed weight on routing plane: first plane reverse:0 on link: 327686 Cost: 6.071100000000002
[MESSAGE]: Progress: 90.0% Current Best: 5.7407000000000004% Perturbation: 6.071100000000002
[MESSAGE]: Changed weight on routing plane: first plane reverse:0 on link: 327684 Cost: 6.0291000000000015
[MESSAGE]: Changed weight on routing plane: first plane reverse:0 on link: 327684 Cost: 5.8233000000000015
[MESSAGE]: Changed weight on routing plane: first plane:1 on link: 1 Cost: 5.811400000000002
[MESSAGE]: Changed weight on routing plane: first plane reverse:0 on link: 262147 Cost: 5.809300000000003
[MESSAGE]: Changed weight on routing plane: first plane reverse:0 on link: 327684 Cost: 5.7918
[MESSAGE]: Final Lap Cleanup...
[MESSAGE]: Undoing unfinished perturbation...
[MESSAGE]: Class0 hop count min: 0.0 max: 4.0 avg: 2.1 stdev: 0.96 num: 224 Util: 8.66 UtilStdev: 8.64
[MESSAGE]: Class1 hop count min: 0.0 max: 5.0 avg: 2.2 stdev: 1.07 num: 276 Util: 8.89 UtilStdev: 8.88
[MESSAGE]: Class2 hop count min: 0.0 max: 0.0 avg: ? stdev: -0 num: 0 Util: 0 UtilStdev: 0
[MESSAGE]: Class3 hop count min: 0.0 max: 0.0 avg: ? stdev: -0 num: 0 Util: 0 UtilStdev: 0
[MESSAGE]: Class4 hop count min: 0.0 max: 0.0 avg: ? stdev: -0 num: 0 Util: 0 UtilStdev: 0
[MESSAGE]: Class5 hop count min: 0.0 max: 5.0 avg: 2.15 stdev: 1.02 num: 500 Util: 8.44 UtilStdev: 12
Mean Delay Class 0: 2.1 Class 1: 2.2 Class 2: ? Class 3: ? Other: ?
Mean Util  Class 0: 8.66 Class 1: 8.89 Class 2: 0 Class 3: 0 Other: 0
[MESSAGE]: Optimisation cycle complete... Please see for possible error messages in the terminal output.
[MESSAGE]: Cost: 5.740700000000004Cost/Uncap: 0.7375000000000042
[MESSAGE]: Job: 19 done!
```

The output shows utilisation and hop count statistics at the beginning and the end for up to 4 traffic classes. The fifth class is the mean of all other classes. At the start of the cycle, these statistics are shown for inverse capacity, unit and random link weights. During optimisation the algorithm outputs information on which link weights were modified and what best current cost value is. Begin and end of perturbations are also indicated. The output provides an overview of the performance and utilisation and hop count values, in order to help identify problems and to monitor progress.

4.2.4 Software Output

Apart from console output, the optimisation software produces several other statistics about the optimisation process as well as network state throughout its operation. Three main categories of information are provided: per link, per demand and per iteration.

Per Link Link information is contained in files taking the format **links_[i, u, r, ipte][x].out**, where i, u, r and ipte stand for inverse capacity, unit, random and ipte weights respectively and the x stands for additional runs done for error calculations. The link files contain a line for each link in the network topology detailing:

- Link ID
- Link bandwidth
- Link load
- One load entry per routing plane.

Per Demand Demand information is contained in files taking the format **demands_[i, u, r, ipte][x].out**, where i, u, r and ipte stand for inverse capacity, unit, random and ipte weights respectively and the x stands for additional runs done for error calculations. Demand files contain a line for every demand pair defined in the demand matrix. The following information is provided:

- Demand ID
- MT-ID or Class
- Number of congested links passed.
- Highest Utilisation encountered.
- Mean utilisation across all links traversed.
- Mean spare capacity on links traversed.
- Standard deviation of utilisation across all links traversed.
- Standard deviation of spare capacity on links traversed.
- A pair of entries for link load and link bandwidth for each link the demand traversed.

Per Iteration Efficiency information is contained in files taking the format **efficiency_ipte[x].out**. There are no equivalents for unit, inverse capacity and random link weights as only IPTE link weight settings undergo the optimisation process. Again the x stands for

additional runs done for error calculations. A line of results is added to the file at the end of each iteration providing the following information.

- Time (ms) since the start of the optimisation.
- Total cost during this iteration.
- Mean cost across all links.
- Utilisation mean across the whole network.
- Cost standard deviation.
- Utilisation standard deviation.

4.2.5 Efficiency and Optimisation

Measuring the algorithms performance is important in order to improve its convergence properties. The heuristic algorithm used for the IPTE system has to search a large solution space that is a function of the number of links and routing planes. The plots in figure 4.2 provide information on the algorithms convergence properties. The top plot shows the improvement of cost function over iteration number plotted on a log scale, whereas the second and third plots track the mean and standard deviation of network utilisation. All graphs feature two plots, the best and the running value.

The running value is the value computed for the current iteration and may be worse than the best value. If the value is worse, the new solution will be discarded at the beginning of the next iteration. The running cost value features large bumps caused by the perturbation feature of the algorithm, which perturbs the link weight set if no improvement is found to the best solution for some time. It is meant to enable the algorithm to escape from suboptimal local minimum solutions. After a perturbation begins, a new temporary best cost value is determined which enables the algorithm to continue optimising the perturbed solution if it is worse than the original solution initially.

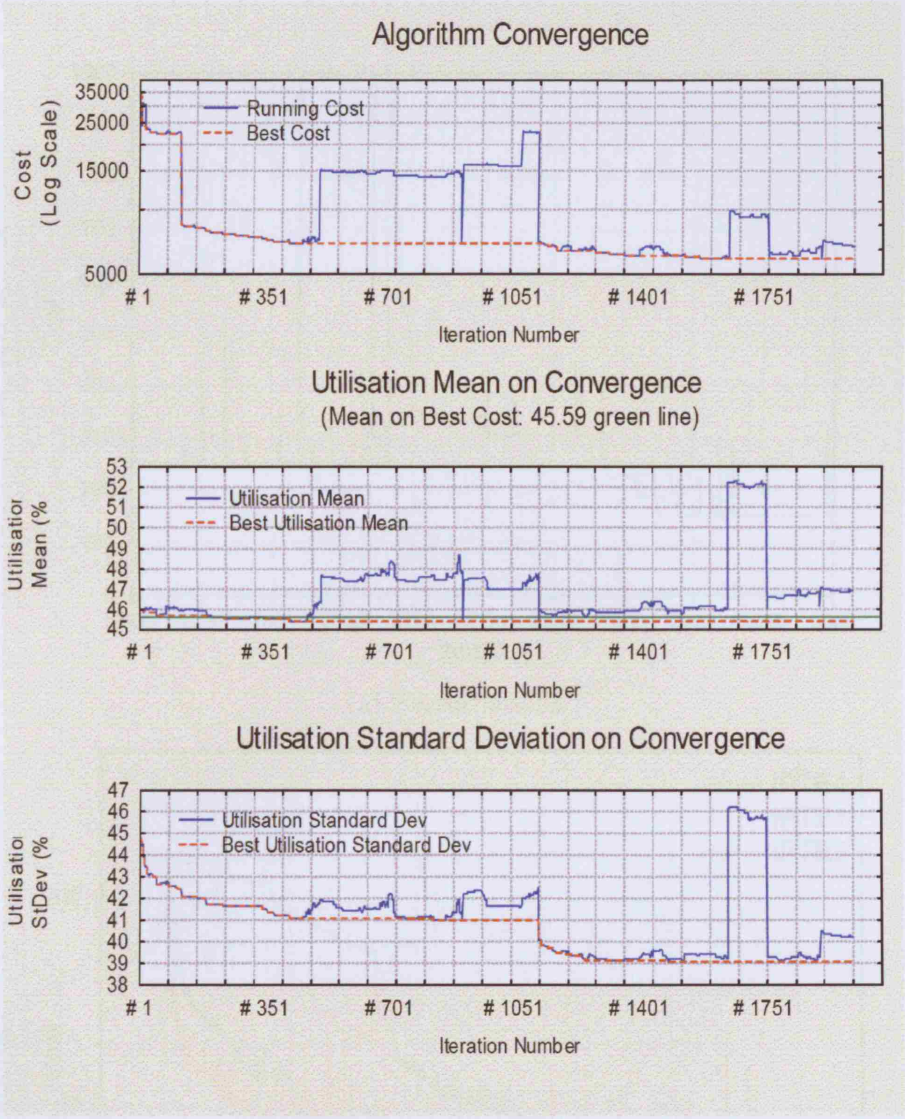
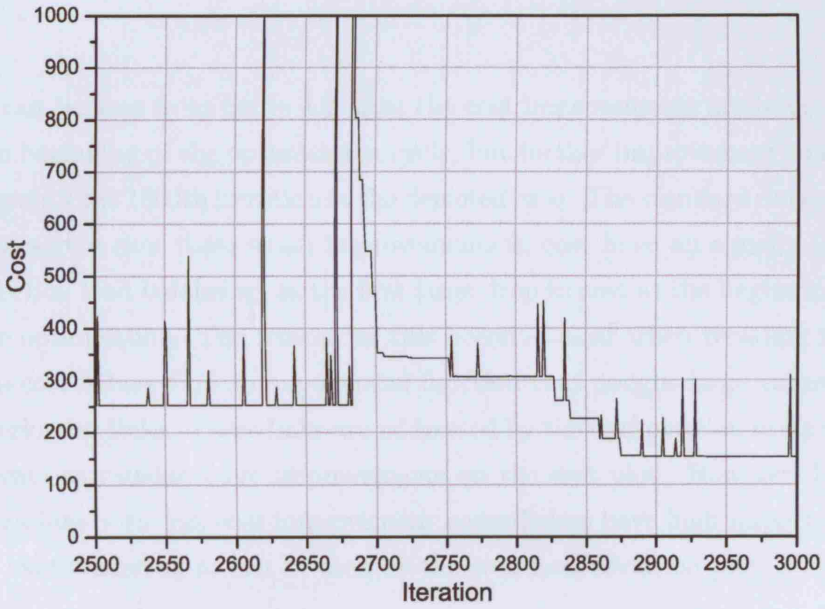
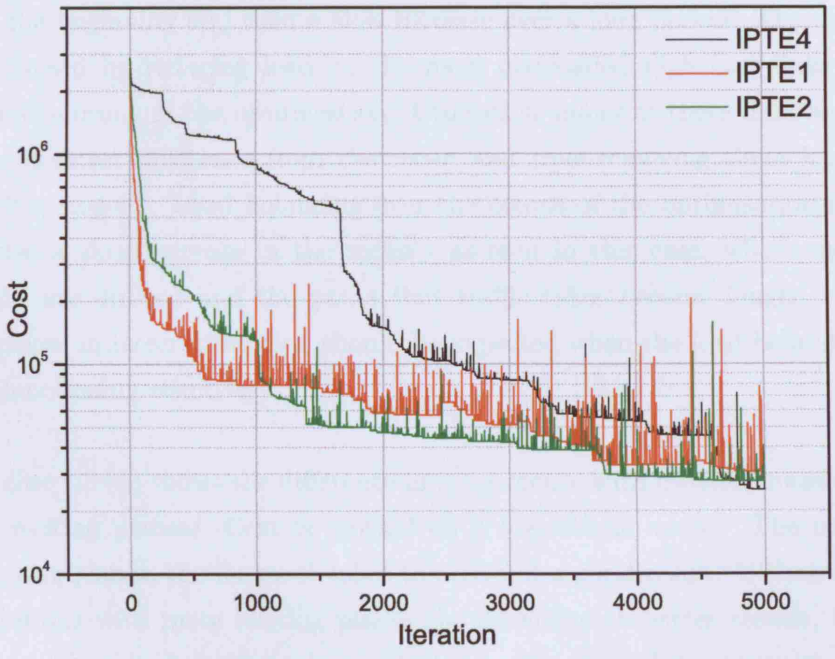


Figure 4.2: Convergence Efficiency

Once a perturbation has completed it is discarded if unsuccessful or kept if an overall improvement was achieved. A magnified view of a successful perturbation is shown in figure 4.3 (a), where the original perturbation caused a large increase in cost, but subsequent link weight optimisation reduced the cost of the perturbed solution below that of the previous best.



(a) Perturbations



(b) Convergence of 1,2 and 4 Routing Planes

Figure 4.3: Perturbations and Multi-plane Convergence Efficiency

As before the spikes indicate single high cost solutions in the running cost value.

It can be seen from figure 4.2, that the cost improvements are largest at the beginning of the optimisation cycle, but further improvement continues until the 1600th iteration in the depicted case. The standard deviation plot shows that these small improvements in cost have an equally large effect on load balancing, as the first large drop in cost at the beginning of the optimisation. The reason for this becomes clear when recalling that the cost is based on an exponential function that assigns large values to overloaded links. These links are addressed by the optimisation in its first iterations causing large improvements on the cost plot. However, later iterations with less cost improvement nevertheless have high importance to load-balancing as can be seen on the standard deviation plot.

The mean utilisation stays approximately constant with a slight decrease in the beginning and then a slow increase over a long period. The drop is caused by reducing load on the most overloaded high cost links at the beginning of the optimisation. Utilisation values of these links show the greatest difference from the mean and thus removing them has a visible impact. Load balancing over the course of the optimisation can cause a slow increase in the mean - as seen in this case, where more links are utilised and the paths that traffic takes become longer. An increase in mean utilisation should be expected when the load balancing is functioning effectively.

Figure 4.3 (b) shows the difference in convergence with different numbers of routing planes. Cost is plotted on a logarithmic scale. The more routing planes, the longer it takes to arrive at a good result. Ultimately, solutions with more routing planes should arrive at better results, but as can be seen from the graph, solutions with one and 2 routing planes arrive at good solutions much faster than the run with 4 routing planes.

4.2.6 Computational Requirements

Fortz and Thorup quote computation times of up to 2.5 hours for 500 iterations with a network of 50 nodes and 212 links for their "non-dynamic" case (for each link weight modification, the cost on the whole network is re-calculated). These results are based on a not defined computer no newer than from the year 2000. Results with similar network sizes with the IPTE algorithm computed on an Intel CoreDuo Mobile CPU with 2.16GHz and 2MB L2 cache take between 30 minutes for 5000 and several hours with 10000 and more demand pairs. The simulators memory footprint depends on nodes, links and demand pairs and is between 500MB and 1GB for network sizes between 50 nodes and 100 links and 100 nodes and 400 links. Since the implementation is based on Java and the impementation of Fortz and Thorup is based in C, a direct comparison is not possible, but it is likely that the use of Java causes an increase in both computing time and memory footprint. In addition to CPU and memory requirements, each simulation run requires up to 50MB of hard drive space to store output data. This amount of storage capacity required depends on the number of iterations (size of the efficiency output), size of network (per link output) and demand matrix (size of the demand output).

4.3 Supporting Tools

Several supporting tools were developed to aid operation and data analysis for the optimisation software, most notably:

Demand Generator The demand generator was written in Java and generates demands as specified in section 3.4. The demand generator is part of the main body of code for the optimisation software and has to be configured manually from within the IPTE main function.

Data Analysis Java Tools A tool was created to collect data for the plots in the results section. The tool also calculates the mean and

standard error across several runs with different topologies or demand matrices. The tool is named average.java.

Data Analysis awk scripts Several awk scripts were written for earlier versions of the software, whose were largely replaced by average.java.

4.4 Summary

This chapter has provided a description of the optimisation software that was designed and implemented as part of this thesis. An implementation was necessary as none of the existing simulation tools provided the necessary functionality to optimise link weight configurations. The implementation is based on the source code of the BRITE topology generator, which provided some useful graph data structures. The chapter outlines the design of the software and provides a description of its operation. The chapter also provides an investigation into the efficiency of the software, since speed is an instrumental part of the optimisation of large network topologies. Finally, additional supporting software tools that were written mostly for data analysis and demand generation were introduced. The source code for all software introduced in this chapter can be found on CD attached at the back of the thesis.

Chapter 5

Link Weight Optimisation Evaluation

5.1 Introduction

This chapter is the second part of the IPTE design and analysis, it presents the methodology for analysis and the simulation results based on the theoretical framework of the last chapter.

5.2 Methodology for Analysis

5.2.1 Introduction

The overall goal of the simulations is to establish whether the IPTE algorithms perform as expected and if they can be used effectively to optimise OSPF networks. The simulations have three objectives:

1. Functional Validation
2. Algorithm Performance Measurement and Optimisation
3. Algorithm Efficiency Measurement and Optimisation

Functional Validation is concerned with ensuring the correct function of the algorithm as well as the simulation software. The outcome of this

testing phase provides some reliability assurance in later test results. The phase consisted of writing testing methods designed to test individual components of the simulator as well as running optimisations on some small test networks that allow by-hand validation of results more easily.

Algorithm Performance Measurement and Optimisation is the essential phase during which the traffic engineering capabilities of the algorithm were measured. Load balancing capabilities of the algorithm were compared with standard link weight distributions. The effectiveness of using routing planes to perform load balancing was analysed.

Algorithm Efficiency Measurement and Optimisation is that part of the analysis dedicated to the algorithms computational properties. Results for these tests were presented in the last chapter and focused on resource consumption (i.e. processing requirements), scalability with topology as well as traffic demands and convergence time.

5.2.2 Network Topologies

Three types of topology were used for simulation: generated Waxman topologies, Rocketfuel inferred real topologies and a small topology from the early NSFNet which is displayed in figure 5.1. The Waxman topolo-

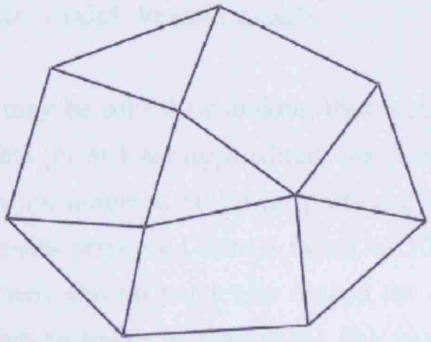


Figure 5.1: Early NSFNet Topology

gies were generated using the BRITE topology generator Medina et al.

[118] with either flat or exponential bandwidth distributions. Exponential bandwidth distributions reflect that the more connected core of an intra-domain network contains higher capacity links than the network edges. Most importantly for simulation, the exponential bandwidth distributions show the effect of the inverse capacity link metrics, while in unit bandwidth distributions inverse capacity equal unit metric assignment. BRITE topologies were used for comparison purposes as well as for their easy availability.

Rocketfuel inferred topologies [119] are included in order to provide simulations on realistic network topologies. However, their size and the resultant simulation time have limited the number of simulations that have been performed.

Some observations and resulting assumptions have to be made before analysing any Rocketfuel inferred topologies. The data obtained through the Rocketfuel probing technique includes links, nodes and link weights. It does not provide measures of link capacity, nor does it allow any inference of traffic patterns on the network. Two problems thus present themselves before an IPTE simulation can be performed:

(1) How to assign capacities to the links
(2) How to generate a valid demand matrix

The first problem may be solved by making that assumption that inverse capacity link weights (or at least approximations thereof) were used, thus the link capacities are assumed to be proportional to the provided link metrics. For the cases presented here a factor of 100 was applied, thus $c(l) = 100/\omega_l$. There was no particular reason for choosing a factor of 100, but it did seem to result in reasonable link capacities in the range of 1-500. Ultimately the capacities (and their units) are irrelevant, since the traffic matrix can be scaled in order to suit any set of capacity values.

The second problem cannot be solved and the previously presented techniques for demand generation are therefore used. As a result of this, it is not possible to claim any improvements or make recommendations for operators of the networks in question, however, their analysis provides some further evidence of the potential that IPTE presents.

The NSFNet topology was used since its small enough to immediately see the effects of optimisation. It was both used as a validation topology and better visualise the effects of IPTE load balancing.

A list of topologies that were used along with more detail of each topology is displayed in table 5.1. Each simulation in the results is based on one of these topologies.

BRITE Generated Waxman Topologies						
Number of		Link Capacity		Distribution		Average
Nodes	Links	Min	Max	Bandwidth	Node	Degree
30	60	10	10	flat	heavy tail	2
50	100	10	10	flat	heavy tail	2
50	100	10	1024	exponential	heavy tail	2
50	200	10	1024	exponential	heavy tail	4
50	100	10	1024	uniform	random	2
50	200	10	1024	uniform	random	4

Rocketfuel Inferred Topologies and NSFNET Topology					
Name	Nodes	Links	Min Capacity	Max Capacity	Degree
NFSNet	10	17	2	5	1.7
Ebone	88	161	1	500	1.82
Exodus	80	147	1	500	1.84

Table 5.1: Topologies used for Simulations

5.2.3 Measurement Technique and Assumptions

Several techniques are in existence for measuring the various properties of networks. Quantities such as number of links and nodes, link capacity and link utilisation are the most basic statistic properties of networks.

Additionally, the node degree describes the number of links connected to a node with indegree and outdegree being special cases for directional links. Links can also have many properties, such as delay, distance and other qualitative measures.

For the purposes of measuring the effectiveness of IPTE optimisation solutions, the author first attempted to use the methods deployed by Forz and Thorup. However, their measurements were mostly based on their respective cost functions rather than properties of the resultant traffic. While measuring how the cost of ones own cost function is large for someone else's solution and much smaller for ones own solution is compelling, it does raise the question of how much can be inferred about the state of the network simply through stating a cost value. The author has therefore decided to take a more practical approach of attempting to calculate traffic flows in the network. These traffic values are estimates, since no packet level simulation or even real deployment has been attempted.

During early testing, it was found that values of average network utilisation and standard deviation of network utilisation are insufficient measures for analysing solutions, they are very aggregated and provide little information on the overall network state. They also do not take link capacity into account and thus more careful differentiation of capacities is required before a mean can be calculated. Furthermore the use of standard deviation is questionable, since a good understanding of the overall distribution of link utilisation is required before analysis of standard deviation values becomes meaningful. For instance, it was found that while some optimised networks (at least superficially) exhibited some approximation to a normal distribution of link utilisation, solutions with inverse capacity and unit link weights tended to have large spikes at zero or beyond 100% and far away from the mean. Instead of using the standard deviation, it was therefore decided to show a histogram of link utilisation for some solutions.

Since all traffic is coexisting on one network, traffic that is part of different routing planes (or classes) is sharing links. In order to derive load statistics, the utilisation figures are calculated for each class individually. For each class, each link containing traffic of the class is taken into account for load calculation resulting in a *per class utilisation.* Likewise, those links that do not contain traffic of a particular class are not taken into account. This means that empty links are not taken into account by any load calculation and thus a measure for *overall network utilisation* is also introduced, which takes into account all links and classes. While the load statistics for individual traffic classes give a measure of experienced network load, the network load statistics give a measure of load balancing. Finally a measure of mean utilisation across all classes gives the mean of all classes excluding empty links.

5.2.4 Limitations

There are several limitations to the results presented here, mostly due to the fact that no packet level results are available, either simulated or from an implementation on an operational network. These limitations are the factors that impact real traffic on real networks: queuing delays, arrival patterns and the effects of non-uniform traffic flows. Also not taken into account are the deteriorating effects due to already experienced congestion. Any traffic trunk passing through a congested link will thereafter be reduced and have a different effect on load of later links. However this is a special case: if the traffic matrix was specified in such a way that congestion is inevitable and even link weight optimisation cannot avert the problem, deployment of such a network is inadvisable.

Generally, this work is seen as a routing study and the effects of traffic flows are assumed to be equal or similar for different link weight solutions. However, if this assumption were incorrect and it were possible to model additional effects, they could be taken into account by the optimisation cost function. Because of the complexity of modelling packet level traffic behaviour, the problem is left for further study.

Additional limitations stem from the network topologies and demand matrices used. Apart from the Rocketfuel inferred topologies, the optimisation is based on artificial network topologies. To limit the impact of this problem, optimisations were based on several different configurations of topologies. Similarly two methods for generating traffic matrices were used to show the applicability of the optimisation algorithm across different matrix types. However, until an optimisation is performed on a production network, results have to be classified as indicative.

5.3 Guide to Results

Load Balancing							
Setup	Topology	Nodes	NP	BWD	RPs	Demands	Degree
1	NSFNET	10	N/A	N/A	64	Flat	1.7
2	Waxman	30	HT	Flat	4	Flat	2
3	Waxman	50	HT	Flat	4	Flat	2
4	Waxman	50	R	U	4	Flat	2
5	Waxman	50	R	U	4	Flat	4
6	Waxman	50	HT	EXP	4	Flat	2
7	Waxman	50	HT	EXP	4	Flat	4
8	Waxman	50	R	U	4	Gravity	2
9	Waxman	50	R	U	4	Gravity	4
10	Waxman	50	HT	EXP	4	Gravity	2
11	Waxman	50	HT	EXP	4	Gravity	4
12	Ebone	88	N/A	N/A	4	Flat	1.82
13	Exodus	80	N/A	N/A	4	Flat	1.84
Keys BWD: Bandwidth Distribution, NP: Node Placement, U:Uniform RP: Routing Planes, R: Random, HT:Heavy Tailed, EXP:Exponential							

Table 5.2: Load Balancing Simulations

Table 5.2 shows all load balancing simulations and table 5.3 shows the configurations that were used for the QoS constrained simulations. Each simulation was run for 5000 iterations per IPTE configuration, i.e. 5000 iterations with each 1, 2 and 4 IPTE routing planes.

QoS Constrained		
Setup	Constraint	Utilisation
2	Bandwidth	30%
2	Bandwidth	60%
2	Delay	30%
2	Delay	60%

Table 5.3: Constrained Simulations

5.4 Functional Validation on NSFNET Topology

This section is a discussion of the optimisation results for load balancing demands on the NSFNet topology, its purpose is to illustrate the some concepts of the process to the reader. Because of the small size of the network, the IPTE simulator was run for 100 iterations each time. In order to better show the function of IPTE, the demands were chosen so that the two of the three core links and some of the smaller links suffer congestion if inverse capacity weights are used. The results in figure 5.2 show the utilisation levels for each individual link. For the given

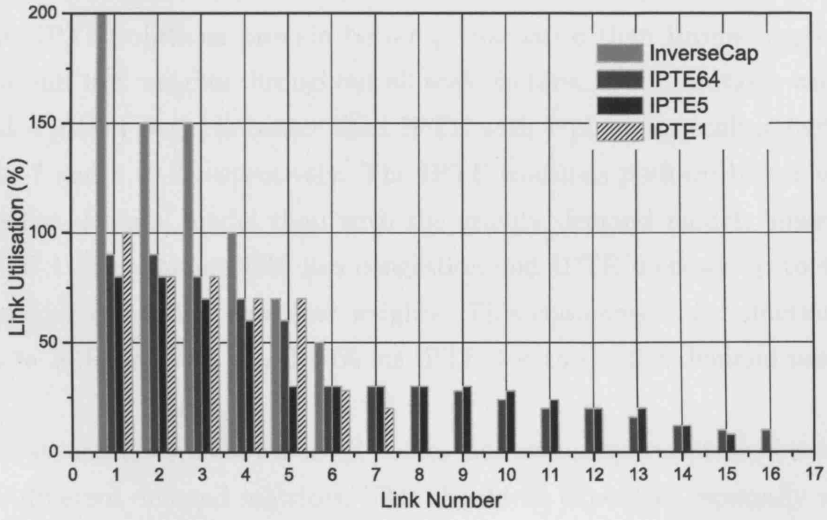


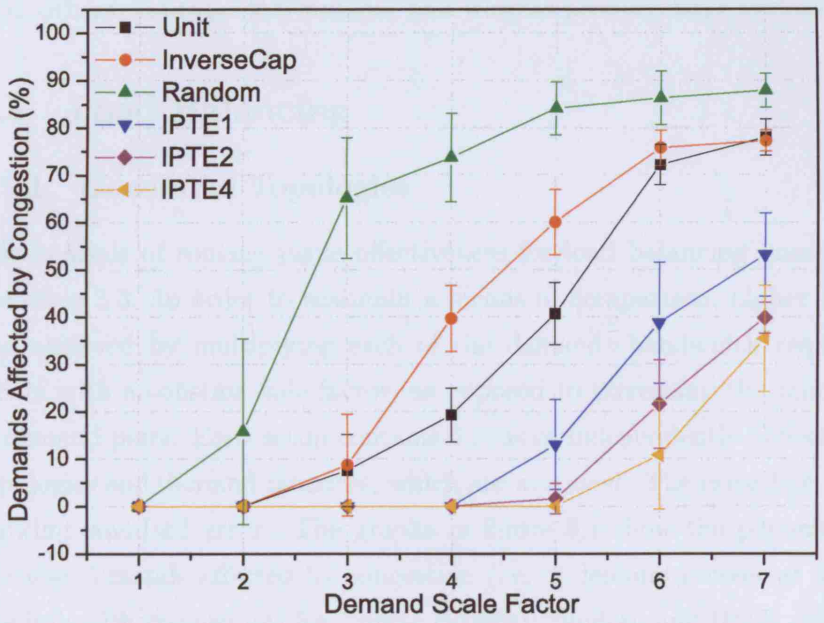
Figure 5.2: Load Balancing Demonstration on Setup 1

demands, IPTE is able to achieve load balancing so that congestion is

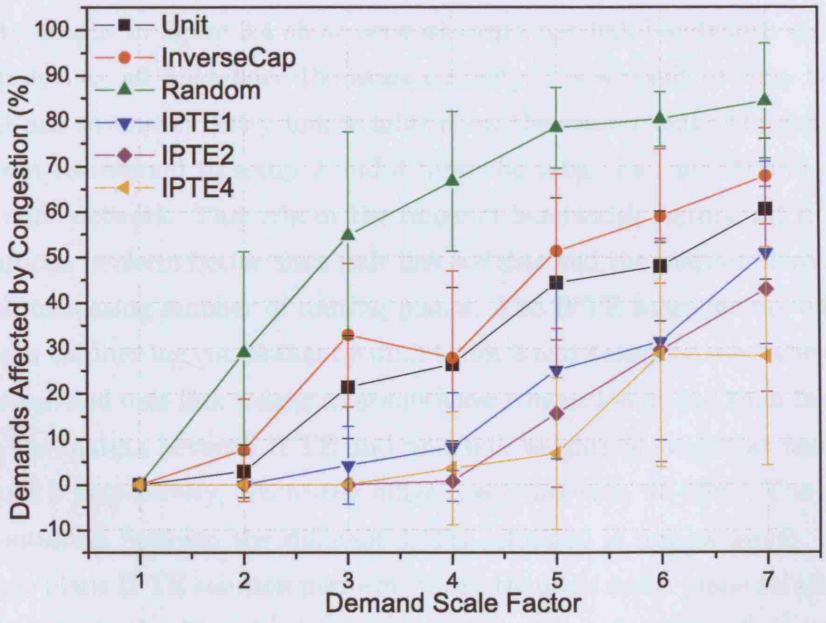
removed from all links. The single plane solution is effective in removing congestion, but it fails to utilise more links and spread the load more equally than the inverse capacity solution, because of the limited number of traffic sources and sinks used. In order to make use of more routing planes, the demands were randomly assigned first to 5 routing planes and then to 64. Five routing planes show a small further improvement on peak utilisation, with clearly visible increase in utilisation of links that were previously not utilised at all. Hence the overall utilisation increases while the peak utilisation decreases slightly. Spreading the traffic over 64 planes shows no further reduction in peak utilisation. The reason for this is the lack of choice of path alternatives given by the network topology.

The two graphs in figure 5.3 show results for the NSFNET topology with both a flat demand matrix and a matrix following the gravity model. Each data point shows the mean over 10 individually generated demand matrices, error bars show standard error. The graph shows how what percentage of demands crosses congested links in each scenario. IPTE solutions are suffixed with a number indicating the number of routing planes deployed for this scenario. It can be seen from the both graphs that IPTE solutions provide better performance than inverse capacity and unit link weights throughout all scale factors. IPTE solutions with 2 and 4 planes perform better than IPTE with 1 plane for scaling factors 5 to 7 and 3 to 7 respectively. The IPTE solutions perform better with the flat demand model than with the gravity demand model, however, IPTE 1 shows about 15% less congestion and IPTE 4 shows up to 40% less congestion than unit link weights. This compares to a reduction of up to 30% for IPTE 1 and 60% for IPTE 4 with the flat demand model.

The standard error shown on all points indicates large variations between the different demand matrices. This should be expected, especially with the gravity model, which generates demand pairs with large variations in bandwidth. Error bars on the IPTE solutions are generally larger than those of unit and inverse capacity link weights indicating that some



Flat Demands



Gravity Demands

Figure 5.3: NSFNET Topology with Flat and Gravity Demands

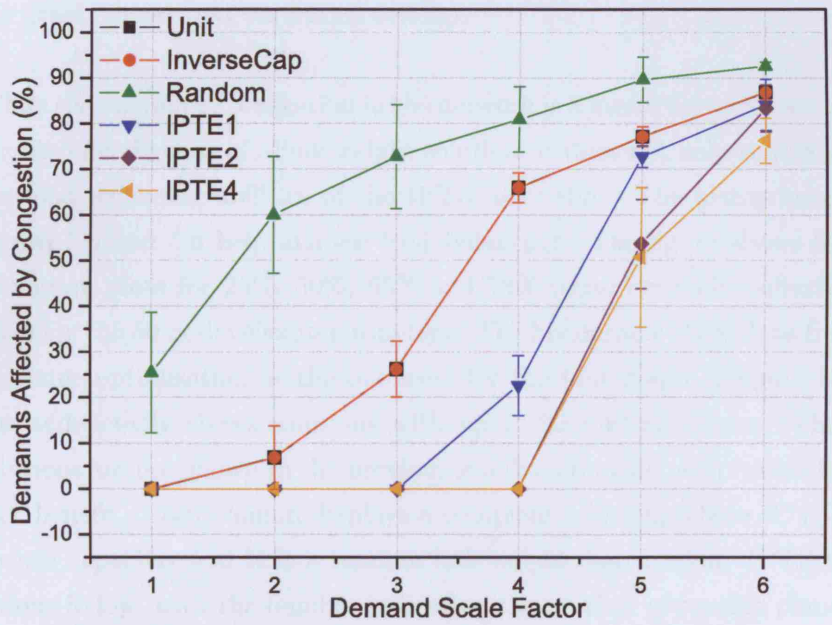
matrices can be accommodated better with adapted link weight settings than others. As expected, random link weights produce large variations.

5.5 Load Balancing

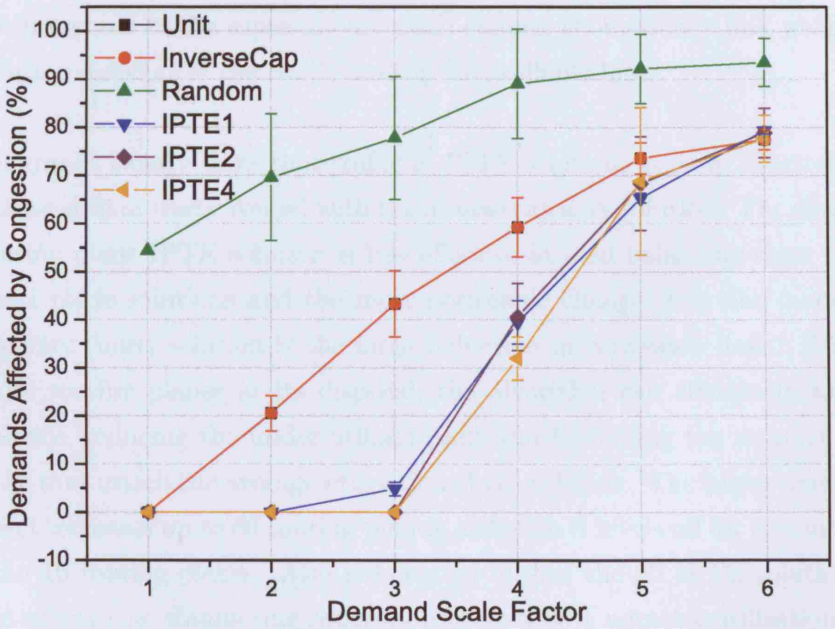
5.5.1 Generated Topologies

The analysis of routing plane effectiveness for load balancing based on the table 5.3. In order to maintain a means of comparison, higher load was achieved by multiplying each of the demands bandwidth requirements with a constant sale factor, as opposed to increasing the number of demand pairs. Each setup contains 5 runs of independently generated topologies and demand matrices, which are averaged. The error bars are showing standard error. The graphs in figure 5.4 show the percentage of these demands affected by congestion (i.e. a demand crosses at least one link with congestion) for inverse capacity, random and IPTE weight assignments.

Both graphs in figure 5.4 show networks with flat link bandwidth distribution, i.e. all links have the same capacity. As a result of this, both unit and inverse capacity link weights show the same result. The graphs shown correspond to setup 2 and 3 from the table, i.e. one 30 and one 50 node network. Throughout the range of bandwidth factors, all IPTE solutions perform better than unit link weights and the margin increases with increasing number of routing planes. The IPTE solutions are effective at eliminating congestion up until factor 3 and 4 respectively, whereas random and unit link weight solutions show congestion at less than factor 2. The margin between IPTE and unit link weights is largest at factors 4 and 3 respectively, where the difference is between 45-65%. The differentiation between the different IPTE solutions is not as large: The single plane IPTE solution performs worse than the multi plane solutions especially in the 30 node simulation, however the advantage of choosing more than 2 routing planes appears to be small. In fact, in the 50 node network case as well as later setup's, IPTE4 solutions don't always per-



Degree 2 – Setup 2



Degree 4 – Setup 3

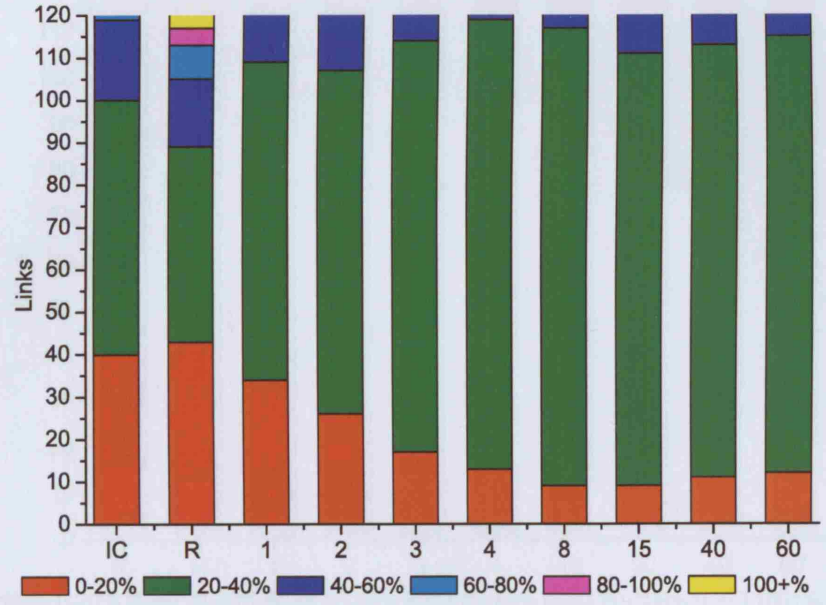
Figure 5.4: 30 and 50 Nodes, HT Flat with Flat Demand Matrix

form as well as IPTE2 solutions. However, the mean error displayed on the graph shows that variations overlap.

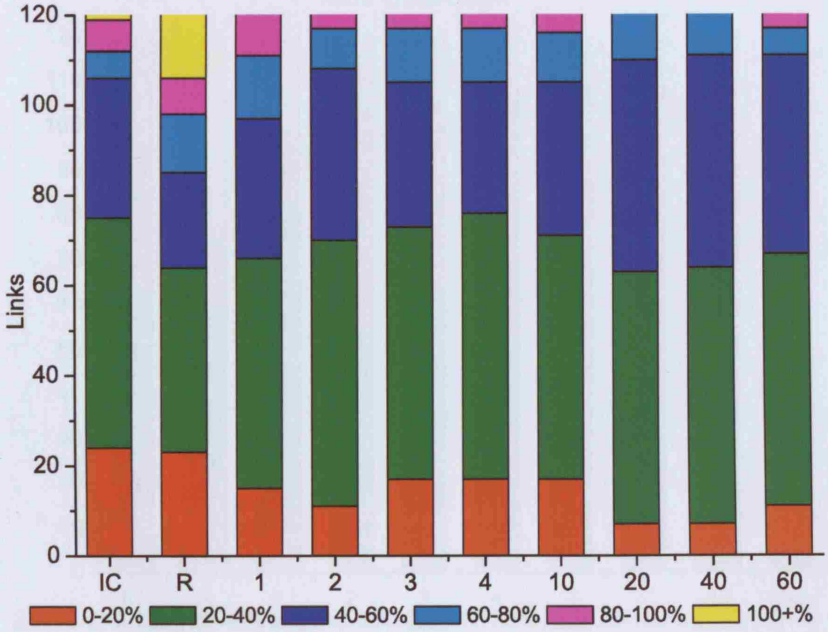
While the amount of congestion in the network is a useful figure for assessing the effectiveness of a link weight solution, it does not help to analyse the load balancing abilities of the IPTE algorithm. The histograms in figures 5.5 and 5.6 help analyse load balancing. The figure shows four histogram plots for 25%, 50%, 65% and 78% mean network utilisation cases for the 30 node Waxman topology. The histograms show data from the same optimisation as the one used for the first graph of figure 5.4, but additionally shows solutions with up to 60 routing planes. These solutions are not shown in the previous graph as they do not provide further benefit. Each column displays a complete solution, where IC is an inverse capacity- and R is a random link weight distribution. IPTE solutions follow, with the number indicating the number of routing planes. Links that are in a particular band of utilisation (from 0% to over 100%) are presented in the same colour. Each column shows how a link weight solution distributes the traffic among the available links.

All graphs clearly show that traffic in IPTE solutions is more effectively balanced than traffic routed with the inverse capacity solution. The single routing plane IPTE solution is less effective at load balancing than the multi plane solutions and the most noticeable change from the inverse capacity (unit) solution is the large reduction in congested links. With more routing planes at its disposal, the algorithm can effectively load balance, reducing the under-utilised links and increasing the number of links that match the average utilisation of the solution. The improvement effect increases up to 60 routing planes, although it levels off for any more than 10 routing planes. Also noteworthy is that the all IPTE solutions are effective at eliminating congestion up to a 65% network utilisation.

Although the results are averaged over five simulation runs with different topologies and demand matrices, an amount of variation between the so-

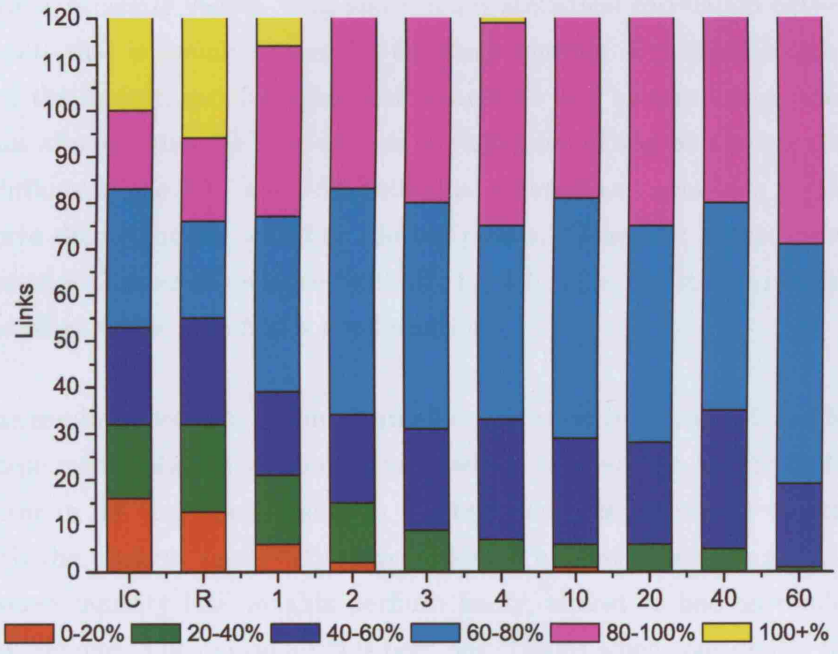


25% Utilisation

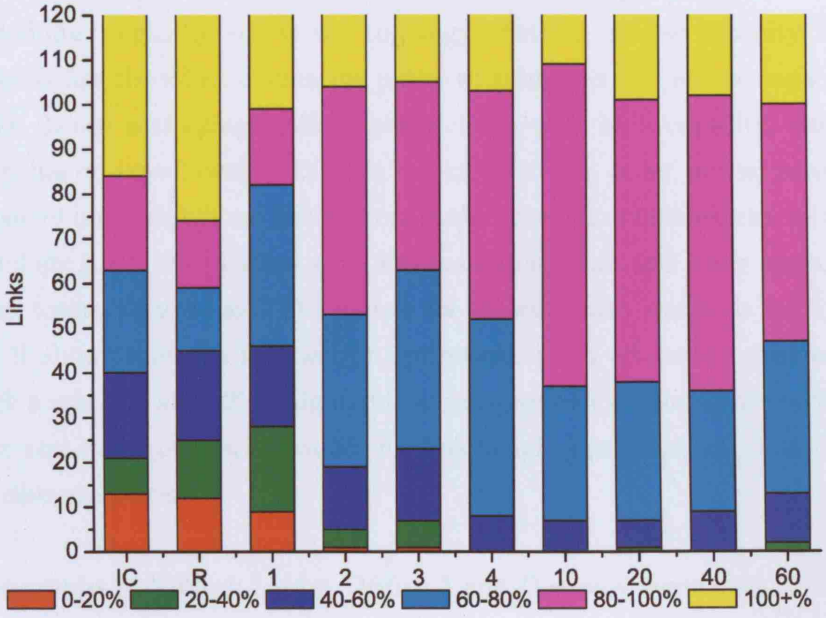


50% Utilisation
for Inverse Capacity (IC), Random (R) and number of IPTE planes

Figure 5.5: Link Load Distribution on 30 Node Network (a)



68% Utilisation



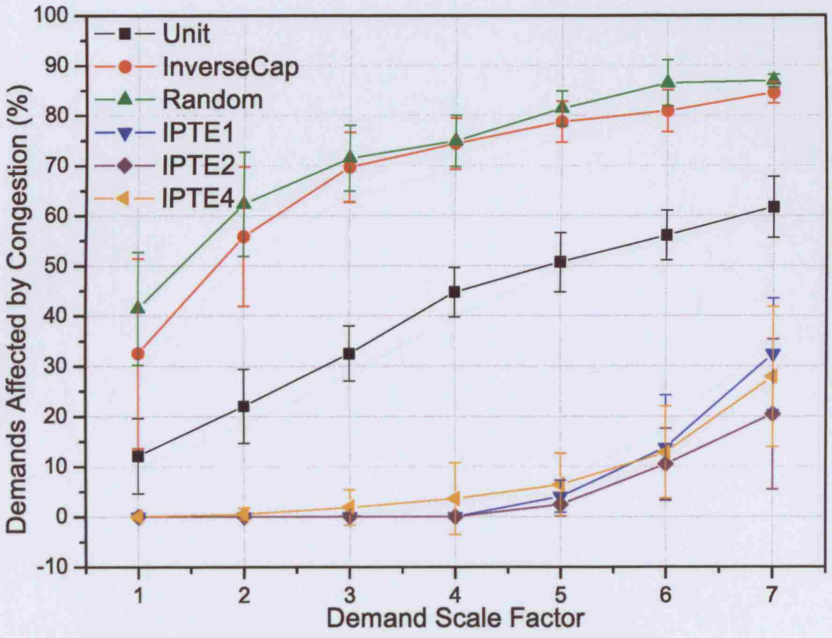
78% Utilisation
for Inverse Capacity (IC), Random (R) and number of IPTE planes

Figure 5.6: Link Load Distribution on 30 Node Network (b)

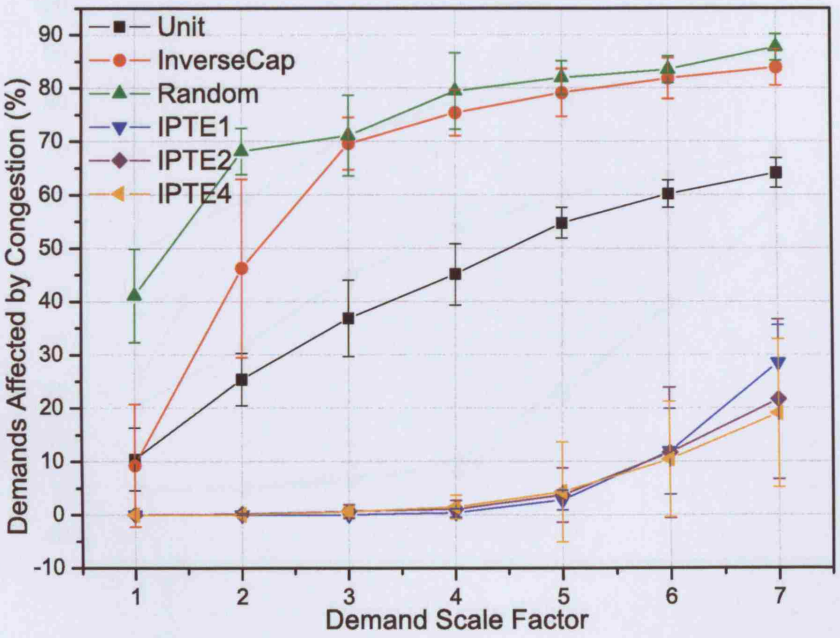
lutions is clearly visible. This shows a low statistical correlation between results that is mainly caused by the large number of possible solutions and the limitations of the heuristic search as well as computing power. This also explains the drop-off that is visible on 40 and 60 routing plane solutions in the 68% and 78% solutions, where these solutions perform worse than solutions with less routing planes. Generally, spreading the demands thinner offers more flexibility for solutions, but it also increases the effort required to find a good solution.

The results of setup's 4, 5 and 8 and 9 are grouped in figures 5.7 and 5.8. These results show a similar pattern, which implies that the overriding factor in defining these results is the random node placement together with the uniform bandwidth distribution. The first observation is that inverse capacity link weights perform badly, almost as bad as random link weights. The reason for this becomes obvious when considering that random node placement means that high and low capacity links are not distributed logically across the topology. Setting inverse capacity link weights has the effect of creating paths with high as well as low capacity links, hence attracting traffic towards links with high capacity, which then has to travel over links with low capacity. In other words, inverse capacity link weights are likely to cause the creation of bottlenecks, as the topology is not constructed with a high capacity core and lower capacity links towards the edges. The reason for showing such results is twofold: (1) it shows that the link weight optimisation can effectively deal with such a scenario and (2) it highlights a dangerous shortcoming of trusting cost and mean utilisation values for link weight optimisation. This will be discussed later.

The graphs in figure 5.7 show Degree 2 and Degree 4 simulation setup's performed with a flat demand matrix and figure 5.8 displays results for the same topologies, but with the gravity model demand matrix. The results show large improvements on all four simulation setup's of up to 55%. Again an improvement can be observed with 2 and 4 routing planes,

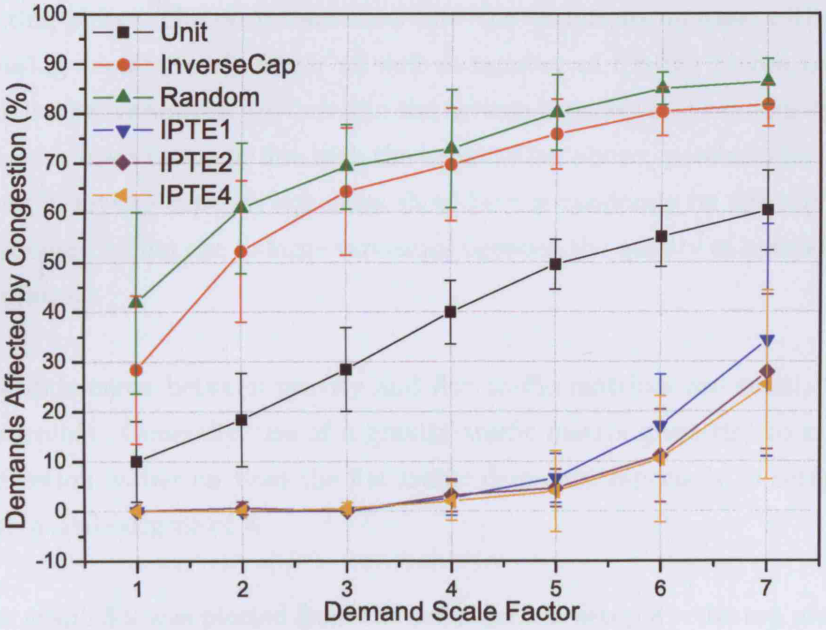


Degree 2 – Setup 4

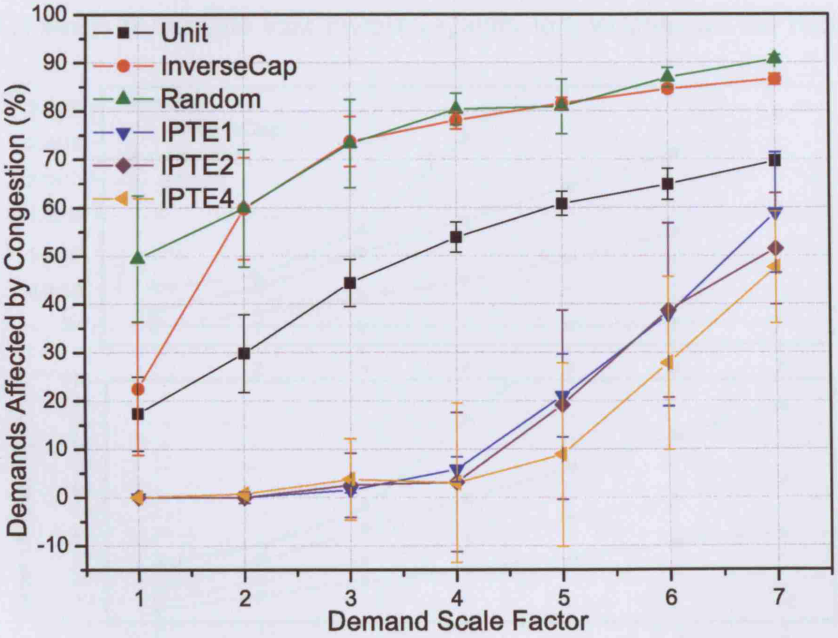


Degree 4 – Setup 5

Figure 5.7: 50 Nodes, R U with Flat Demand Matrix



Degree 2 – Setup 8



Degree 4 – Setup 9

Figure 5.8: 50 Nodes, R U with Gravity Demand Matrix

although not as large as between unit link weights and IPTE with one routing plane. The error bars show how the variations increase with increasing demand scale factor as well as number of routing planes used. The largest variations are found in the inverse capacity and random solutions, an observation in line with the explanation above: potential bottlenecks in inverse capacity solutions should occur randomly on the topologies, hence giving rise to large variations between the quality of individual solutions.

The differences between gravity and flat traffic matrices are small, but discernible. Generally, use of a gravity traffic matrix gives rise to more congestion earlier on than the flat traffic demands, especially in setup 9 with a node degree of 4.

The graph 5.9 was plotted from the same data as setup 8 – the top graph in figure 5.8. It shows some peculiar discrepancies from its counterpart, for it seems to indicate that inverse capacity link weights are the better

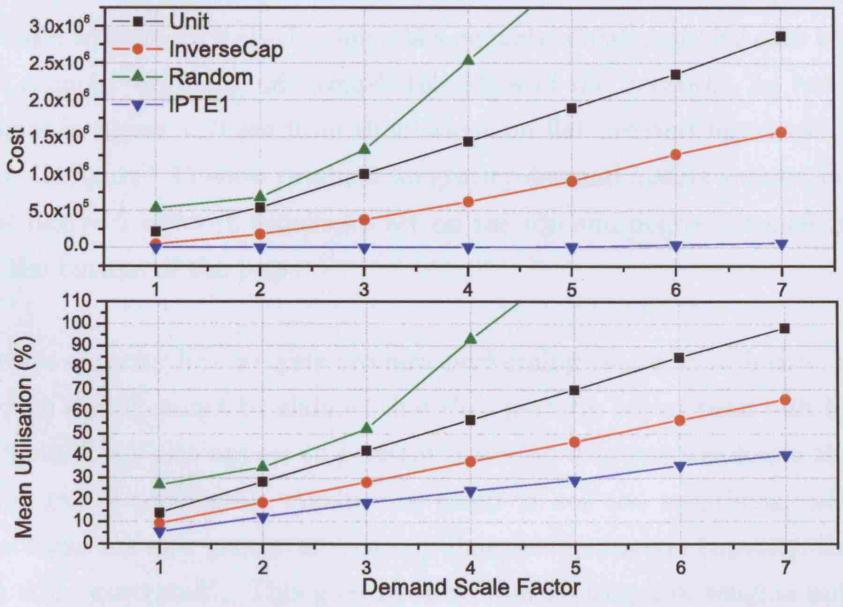
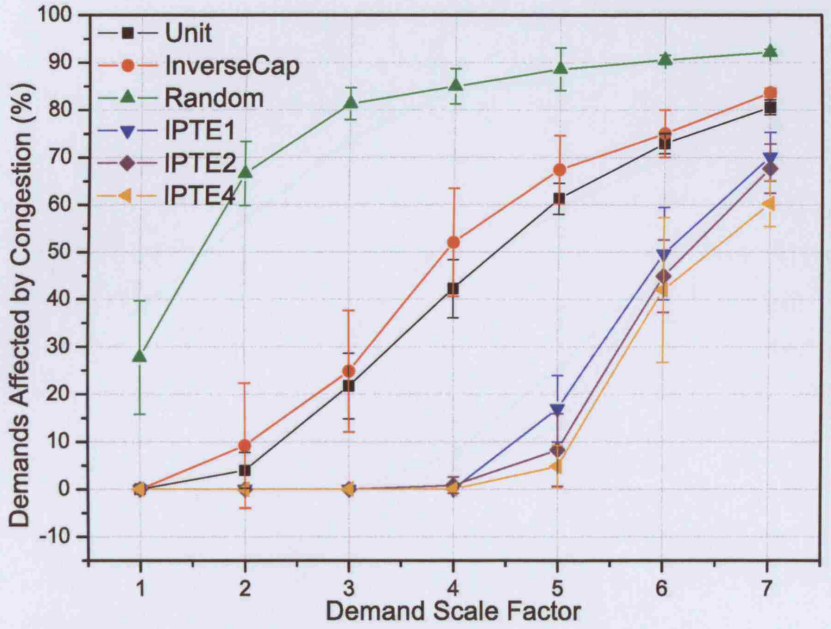


Figure 5.9: Cost and Mean Utilisation for Setup 8

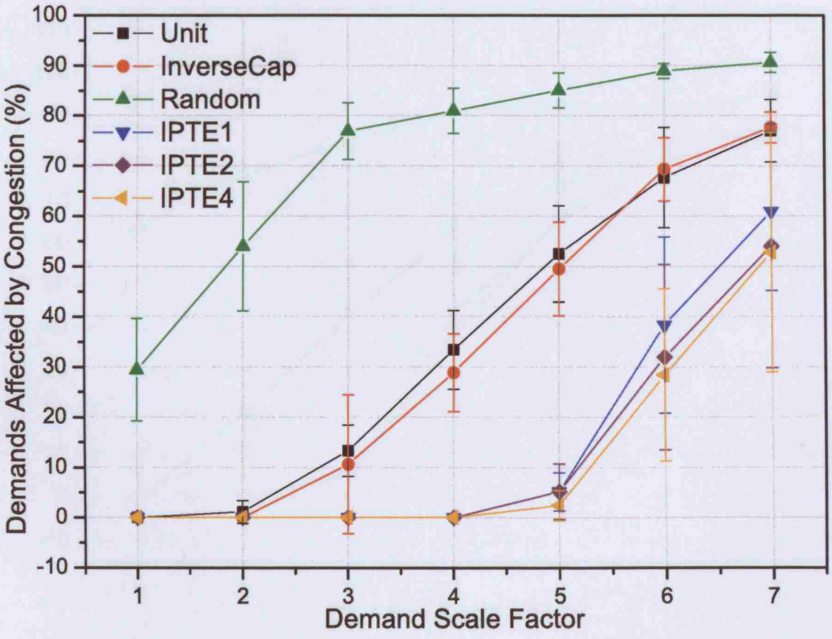
solution compared to unit link weights. The top graph shows cost as calculated from the cost function (3.4) and the bottom graph shows mean utilisation over all traffic. Observations of this type gave rise to the points made in section 5.2.3 and caused the author to look for other means of determining the quality of results than cost and mean utilisation. The reason for the apparent good performance of inverse capacity is that the cost function measures cost based on links and not on demands. It also does not take link capacity into account and thus all congested links cause equal cost. However, a high capacity congested link gives rise to a larger number of affected demands than a low capacity congested link. Thus, an inverse capacity solution can perform well in terms of cost and utilisation, but badly in terms of actual network performance. A better way to cost standard link weight settings would therefore include a cost scaling based on link capacity.

The graphs in figures 5.10 and 5.11 show simulation setup's 6, 7, 10 and 11, corresponding to the network topologies with a heavy tailed node placement and exponential link bandwidth distribution. These topologies are said to be more realistic, since they emulate a high capacity core with the capacity dropping off towards the edges of the network. As before results in figure 5.10 are from simulations on flat demand matrices and those in figure 5.11 show results from gravity demand matrix simulations. The degree 2 network topologies are on the top and degree 4 topologies on the bottom of the page.

Inverse capacity link weights are now performing better than before, although it still cannot be claimed that they perform better than unit link weights. They also appear to perform better on degree 4 topologies than on degree 2 topologies. Again, the result is not too surprising, while topologies are now generated in a way that favours inverse capacity, they are still "generated". This goes to show that setting link weights uninformed and by recommendation can lead to bad performance. A close look at the network topology is imperative before making decisions.

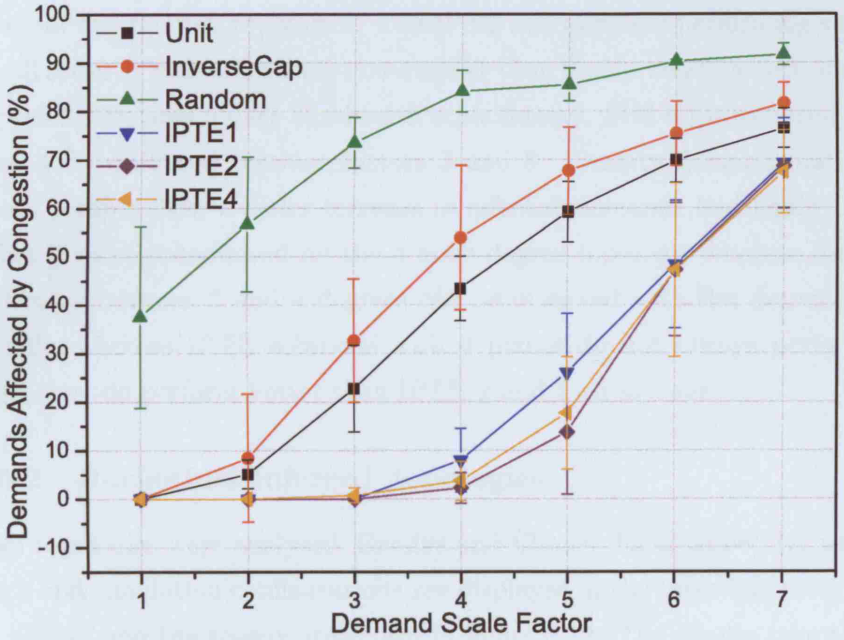


Degree 2 – Setup 6

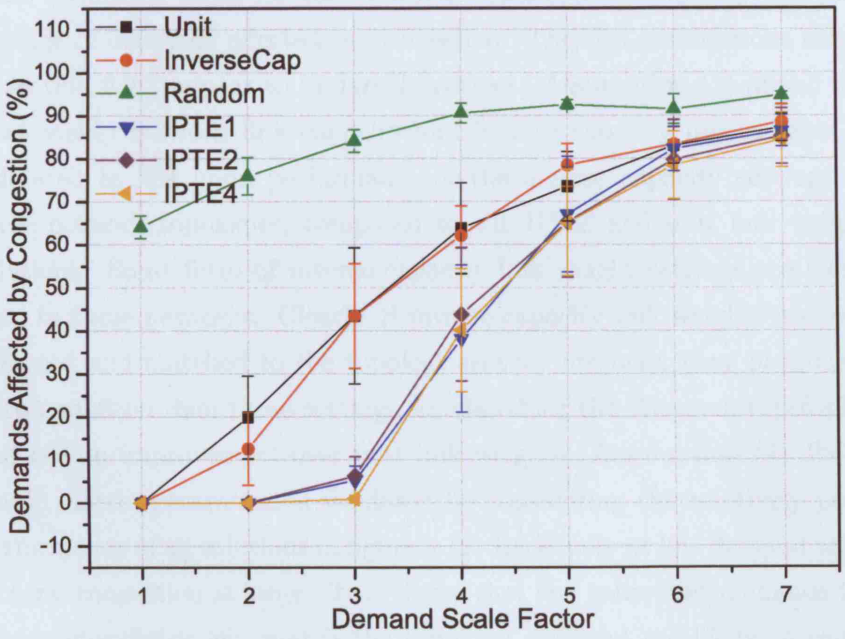


Degree 4 – Setup 7

Figure 5.10: 50 Nodes, HT EXP with Flat Demand Matrix



Degree 2 – Setup 10



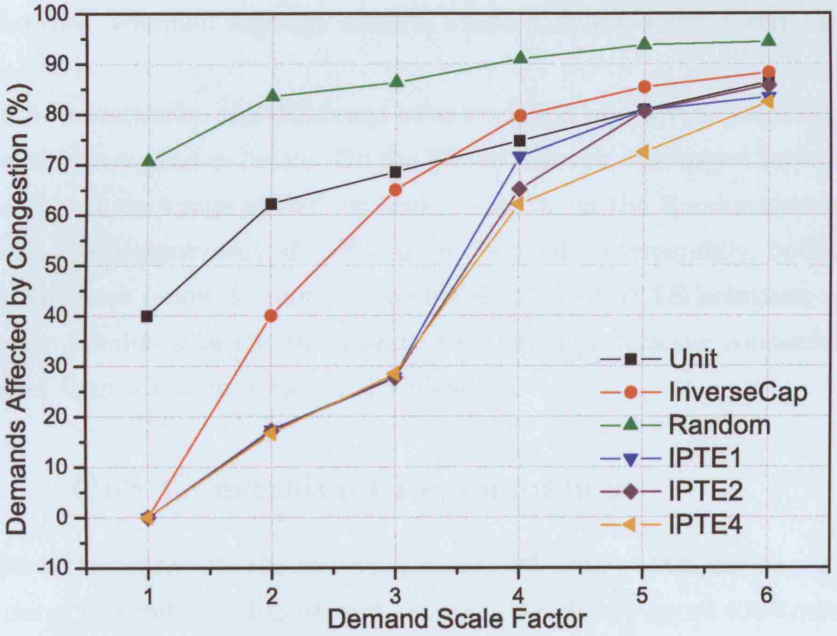
Degree 4 – Setup 11

Figure 5.11: 50 Nodes, HT EXP with Gravity Demand Matrix

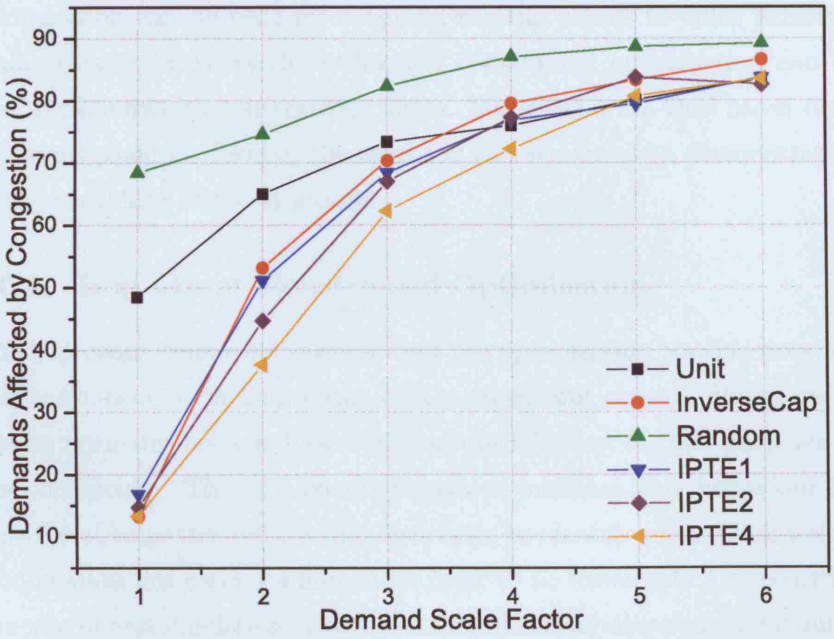
The observations concerning IPTE solutions are much the same as they were for the random topologies. The IPTE solutions are performing well on all setup's, with differences now smaller than for the random networks, especially towards higher bandwidth scale factors. Still improvements of over 40% are found between factors 3 and 5. Gravity demand matrix based setup's show a faster increase in affected demands and again the effect is most pronounced on the 4 node degree topology, whereas little difference between 2 and 4 degrees can be observed with flat demands. Finally, whereas IPTE solutions with 4 planes do not always perform best, they do perform better than IPTE 2 and 1 on average.

5.5.2 Rocketfuel Inferred Topologies

Two topologies were analysed, Exodus and Ebone. Their respective network and simulation configurations are displayed in the table 5.3, setup's 12 and 13 and the results are shown in figure 5.12. The results take the same format as those for the Waxman topologies and display the percentage of demands affected by congestion. The two assumptions made in section 5.2.2 appear to be largely correct. Assumption (1) about the relationship between link capacity and inverse capacity link weights is validated by the good performance of the inverse capacity settings in these network topologies, compared to the IPTE and unit link weight solutions. Some form of inverse capacity link weight settings are likely used in these networks. Clearly, if inverse capacity link weights are well designed and matched to the topology as they are here, they produce a better solution than those settings simulated for the Waxman topologies and are an improvement over unit link weights. Assumption (2) about traffic matrix generation is validated by considering the relatively poor performance of all solutions in figure 5.12. Especially at low demand scale factors, congestion is large. This shows that the generated demands for these networks do not match the topology well and are likely to cause congestion that cannot be avoided though any modification of link metrics. Because of the good performance of the inverse capacity link metrics,



Ebone – Setup 12



Exodus – Setup 13

Figure 5.12: Rocketfuel: Ebone and Exodus Networks

they were used as initial conditions for the IPTE optimisation unlike in the other, Waxman topology setup's, where unit link weights were used.

On both networks, the difference between IPTE and inverse capacity is not as pronounced as before. On the Ebone network, the largest improvement is in the range of 30% on scale factor 3, on the Exodus network, much less improvement of 15% can be observed. Interestingly, both solutions show larger margins between the different IPTE solutions and on the Exodus network, the 4 plane solution is performing consistently better than solutions with 1 and 2 planes.

5.6 QoS Constrained Performance

This section presents the results of constraint based optimisation. Two types of this individual treatment were simulated: hop count constrained and utilisation constrained routing planes. The hop count constrained optimisation can be used for mapping routing planes to delay sensitive traffic classes, whereas the utilisation constrained optimisation can be used for bandwidth constrained classes. More elaborate QoS based optimisations could be devised, the results in this section are a demonstration of the feasibility of the approach.

5.6.1 Hop Count Constrained Optimisation

The hop count constraint ensures that the optimisation routine does not create solutions with long paths on the constraint classes. Paths on all classes normally become longer during optimisation for the purposes of load balancing. The hop count constraint punishes this behaviour for class 3 and helps the optimisation heuristics to identify when a link weight modification has caused a hop count limit to be exceeded. The resulting increase in cost should lead to a discarding of this modification in favour of one that performs load balancing on other classes. As such, it is important to provide sufficient classes to the simulator if load balancing is also an optimisation goal.

The graph 5.13 shows the effect of hop count constrained optimisation. This simulation used the same parameters as that in load balancing setup 10 from table 5.3: a 50 node Waxman topology, node degree 2 with heavy

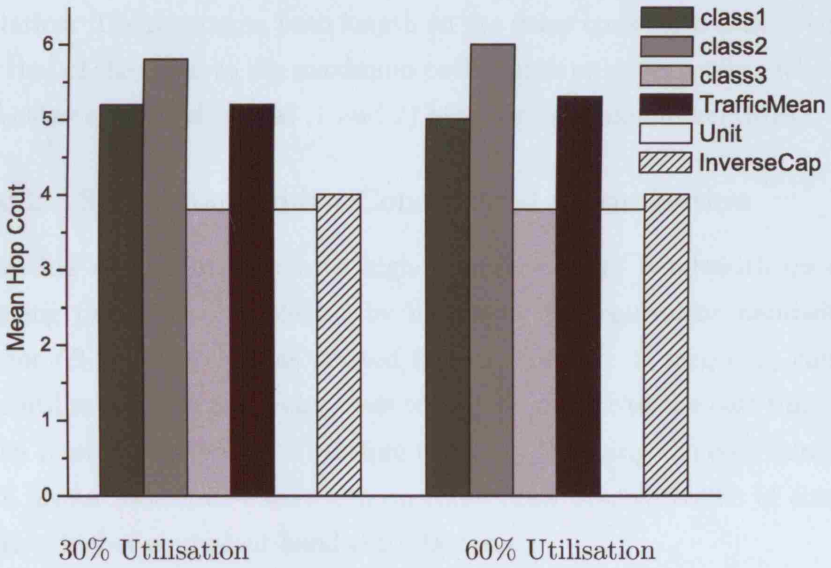


Figure 5.13: Average Hop Count

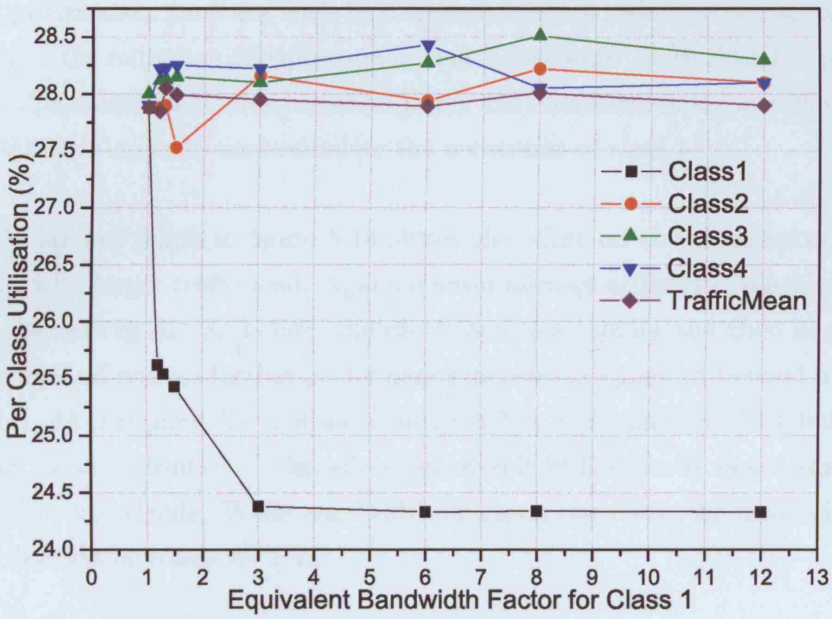
tailed node placement and exponential bandwidth distribution. The demand matrix uses the gravity model. A hop count constraint was applied to class 3, using a sharp maximum hop count cut-off of 6 for class 3 (i.e. a sudden increase in cost from 0 to 5000 between 6 and 7). The maximum shortest path distance across the network is 8 hops. Classes 1-5 (including the constraint class) are operated in parallel on the same network, whereas the results for unit and inverse capacity link metrics are computed separately for the same network topology and demand matrix. Several observations can be made from this graph. Firstly, the hop count for all classes remains approximately the same for different network load. This should be expected, for all reasonable loads, since the cost on any one link should increase evenly with respect to other links and thus higher utilisation should not lead to substantially different solutions. Secondly, it can be seen from the graph that the non-delay constrained classes have high average hop counts. This is also as expected, since longer aver-

age paths are an effect of the load balancing on these classes. Finally, the graph shows that the delay constraint class has a lower average hop count than the non hop-count constrained classes. The delay class performs approximately the same as the unit link metrics, which represents the ideal solution. The maximum path length on the delay constraint class is equal to that of the equal to the maximum path length on unit metrics, whereas all other optimised classes (1 and 2) have larger maximums (10-14).

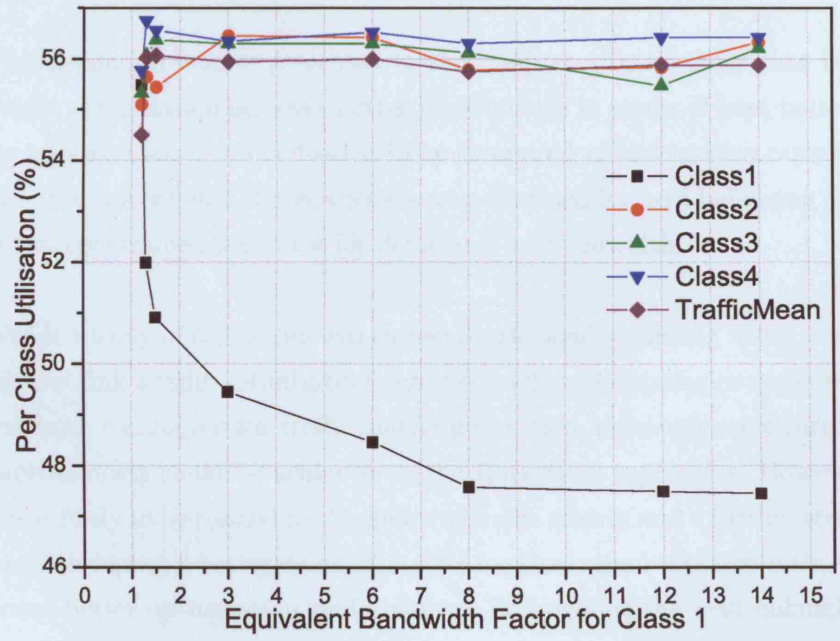
5.6.2 Spare Bandwidth Constrained Optimisation

Creating a configuration with higher average spare bandwidth on one routing plane is accomplished by increasing the equivalent bandwidth factor (β_h) of the class as derived in section 3.6.7. A larger β_h causes the utilisation of a particular class to appear greater to the cost function than it would otherwise. Therefore in theory, reducing the cost function will favour more free bandwidth on those links bearing traffic of classes with a higher equivalent bandwidth factor.

The graphs in figure 5.14 shows the effect of this method. The optimisation was performed on a Waxman 30 node network with uniform bandwidth distribution and with 60 routing planes as per load balancing setup 2. 10000 uniform demand pairs were spread randomly across the network. The graph shows the mean utilisation of all traffic in the network and the mean utilisation of class 1, which is modified by the equivalent bandwidth factor. Three other classes are also shown to demonstrate that classes do not vary widely in bandwidth and that the utilisation of class 1 is not a random occurrence. β_h is shown on the x-axis while the y-axis shows utilisation. With all classes co-existing on the same network, class 1 has lower mean utilisation than all other classes, further decreasing with increasing β_h. The graph shows that the effect first increases rapidly and then levels off when the factor reaches 3. At this point, the final total cost begins to increase again and the optimisation routine is no longer able to balance the loads. At its lowest, the utilisation of class 1 is 12.5% smaller than the mean. While the utilisation of class 1 falls, the average



Low Utilisation



High Utilisation

Figure 5.14: Equivalent Bandwidth Factor

link utilisation for all other classes increases slightly by 1.3% and stays approximately constant with further increasing β_h, which is not surprising as the reduction of utilisation in class 1 levels off. Since class 1 is part of this mean value, this seems to imply that load balancing is suffering slightly in order to accommodate the constraint of class 1.

The second graph in figure 5.14 shows the effect on the same network, but with larger traffic load. Again, a lower average utilisation is achieved by increasing β_h. As before, the effect increases rapidly and then begins to level off and no further performance increase is apparent beyond a β_h of 8. At this point the utilisation of class 1 is approximately 16% below the mean utilisation. The effect on overall utilisation is again small, but clearly visible. While bandwidth drops on one class, the mean class utilisation increases by 1.5%.

5.7 Summary

The simulation results presented in this chapter give evidence that link weight optimisation achieves better performance in terms of load balancing and avoidance of overloaded links compared to the inverse capacity and unit link weights. Simulations were performed for load balancing and in two constrained scenarios for delay and for bandwidth.

A wide variety of topologies was chosen for the load balancing tests, showing that link weight optimisation performs well on all topologies under test and both techniques for traffic matrix generation, although some further improvements could be achieved on the Rocketfuel topologies. However, this is likely to be related to the choice of traffic matrix and a further study into developing a better demand matrix for Rocketfuel topologies should reveal better optimisation performance. Throughout the load balancing simulations, the use of multi topology optimisation was shown to provide advantages over single plane optimisation, although the differences were not as large as those between unit or inverse capacity link weights and

the single optimised routing plane. This is in line with observations made in the literature. IPTE shows the largest advantage over unit and inverse capacity link weights in random topologies, which do not suit standard link weight settings. For these networks improvements of over 55% were observed. The second set of topologies which better model real networks still show improvements of over 40%. The use of more topologies is useful especially at larger scale factors and higher network load, where more topologies show improvements over the single optimised topology of up to 10%.

The observation was made that inverse capacity link weights may not perform as well as is commonly understood, unless the topology is carefully designed, which can be seen in the simulations based on Rocketfuel topologies. While cost and mean utilisation for inverse capacity link metrics on Waxman topologies were consistently lower than unit link weights, the inverse capacity link weights caused more congestion to individual demands. The reason for this is the cost function, which is based on per-link utilisation. While it is a function of utilisation, but does not take into account varying link capacities, and thus equates congestion on low- and high bandwidth links. While this has less bearing on the optimisation, it shows that it is insufficient to rely on cost for evaluating the quality of a solution.

Results were presented for delay constrained optimisation, which show that as long as link delay values are available, they can be used to create one or more routing planes that are optimised based on delay, while other routing planes without delay constraint can be used to perform load balancing. Using a weighting in the cost function, a trade off can be made between delay and load balancing on the delay constrained routing plane, in order to avoid solutions that are very unsuccessful in terms of load balancing.

The spare bandwidth constraint was also simulated and it was shown that using the bandwidth factor, traffic on a single routing plane can be forced onto links with less utilisation. This is useful for provisioning for future demands and it can also serve as a QoS enabling mechanism, since reducing utilisation on the whole path will lead to less queuing delay, jitter and less packet loss.

Overall, the algorithm was shown to work as expected. Using the multiple routing plane approach, networks can be configured with differentiated routing to accomplish a number of optimisation goals ranging from load balancing to QoS based routing. The offline nature of the approach ensures that the network will continue to function as a normal IGP routed network, with no dynamic adaptations for QoS that may cause instability. Of course, this lack of awareness to changes means that IPTE is not a complete solution to load balancing or QoS routing. The next chapter deals with the integration of algorithms like IPTE into a management platform in order to add this missing link.

Chapter 6

Managing Multiservice IP Networks

6.1 Introduction

This chapter combines some of the ideas already discussed so far, to form an extensible and flexible framework for managing multiservice IP networks. While the IPTE approach can be used as a tool for performing sophisticated traffic engineering, it can only show part of its potential if deployed standalone. This chapter will discuss how the IPTE algorithms can be integrated to form part of a powerful and extensible network management system.

6.2 Motivation

> Contemporary Internet networks have three significant characteristics:
> (1) they provide real-time services
> (2) they have become mission critical
> (3) their operating environments are very dynamic.
>
> *D. Awduche [49]*

The graph in figure 6.1 shows how Internet traffic varies throughout a working day. Some useful observations can be drawn from this: firstly, there are large variations in the amount of traffic that flows throughout the day and secondly, there are some clear correlations with daily human behaviour, in other words the traffic variation is not random. The main peak can easily be identified as daily activity. Also visible is a dip at lunchtime and in the evening. At night, most remaining traffic can be attributed to peer to peer transfers and inter-domain traffic from different time zones. Weekly variation can also be observed, most obvious is the higher Sunday traffic. Although there are many more types of traffic

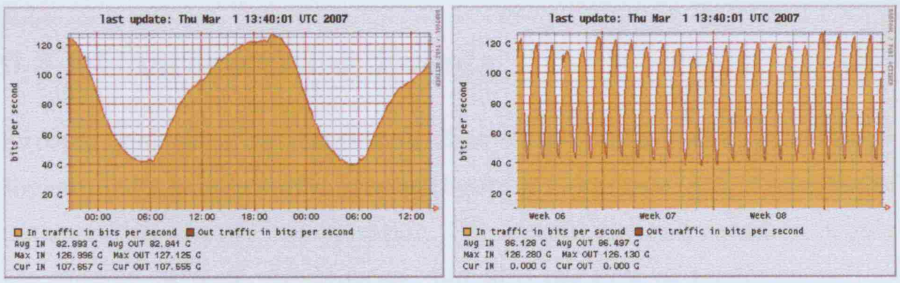


Figure 6.1: Traffic in the DE-CIX Interchange, February 26 2007 [4]

variation, it is immediately obvious that network traffic is more variable than a network operator might like it to be. And since network operations is on a budget like all other departments of an organisation, it is well advised to analyse the phenomena behind these variations in detail, so that the expenses can be kept in check. A number of questions need answering. For example: what is the peak amount of traffic that the network has to carry? What part of the traffic is mission critical and where on the network does it flow? What part of the traffic is low priority? Different questions can be asked for different networks, each with its own set of operational objectives and policies. Once the correct questions have been asked, key objectives can be determined and the task of allocating network resources can begin.

The discussion shows that a well designed management system is needed to maintain a multiservice network. It is not sufficient to install network equipment and hope that it will perform the task. The use of an intelligent network management system can reduce this over-engineering through smarter use of the existing resources. A network management system can maintain state of the many service requirements, predict problem patterns and react by appropriately reconfiguring the network to keep the services operational. The remainder of this chapter is dedicated to integrating the IPTE approach with such a system and to show how it can be used alongside other approaches.

6.3 Characterising Events

Further analysis of traffic reveals that simple daily traffic patterns as shown in the simple traffic graphs (figure 6.1) are only one dimension of patterns that require consideration. For a multiservice network, not only the amount of traffic is changing, but also the type of traffic as well as its source and destination. Whereas office traffic might amount to web research - emails with attachments and corporative work - the bulk of the private traffic in the evening has a large peer-to-peer component as well as web surfing. Taking into consideration some of the not yet existing QoS enhanced services, yet more complex service patterns may occur, VoIP during office hours and Video on Demand (VoD) during evening hours. Other causes of changes in traffic patterns are surges in demand during TV game shows, equipment failures, catastrophic events, etc. The number of causes for traffic surges are large, therefore table 6.1 attempts to introduce some categories by dividing patterns into those that are predictable and those that are not predicable.

Events categorised in the table include:

- Predictable patterns
  - Yearly demand changes

	predictable	unpredictable
traffic	daily -, weekly -, annual patterns	flash crowds, disasters
network	maintenance	failures

Table 6.1: Events Affecting Network QoS

  - Public holidays, Weekends, etc.
  - New applications
  - Office hours
  - Windows security update
  - Network equipment installation/removal
  - Scheduled downtimes for other reasons

- Unpredictable Patterns:
  - Link/Node failure
  - Attack on a network node (e.g. Denial of Service (DoS))
  - Virus or Worm
  - Flash crowd[1]
  - Extraordinary news
  - TV Show with interactive element

The more complex the set of objectives the more complicated the process of engineering the network for all eventualities becomes. Taking into account all or most variations might require a network that is over-engineered only to be able to accommodate demands of a particular time of day or a particular service.

[1]Flash crowd: An event triggered by a news bulletin that causes a large number of people to visit a particular website often resulting in a slowdown or brake down of the website due to the extraordinarily high demand.

6.4 Multiservice Network Traffic Management

Figure 6.2 is based on both the MESCAL (figure 2.4) and AGAVE functional architectures. It shows a more compacted view of the architectures proposed and reduces them to the basic components required for managing traffic in multiservice IP networks. As the MESCAL architecture, this diagram is based on the TMN model. Operations in components near the bottom are invoked more frequently and have a more local view of the network than operations inside components near the top of the diagram. In accordance with this, the management system in this diagram shows three vertical separations. Control plane operations are housed at least partially within network nodes and can contain routing and admission control as well as other dynamic adaptive behaviour configured by the operational management components. Operational management is concerned with configuration and optimisation of available network components as well as customer subscriptions. Long term planning contains the high level service and network design as well as policy control.

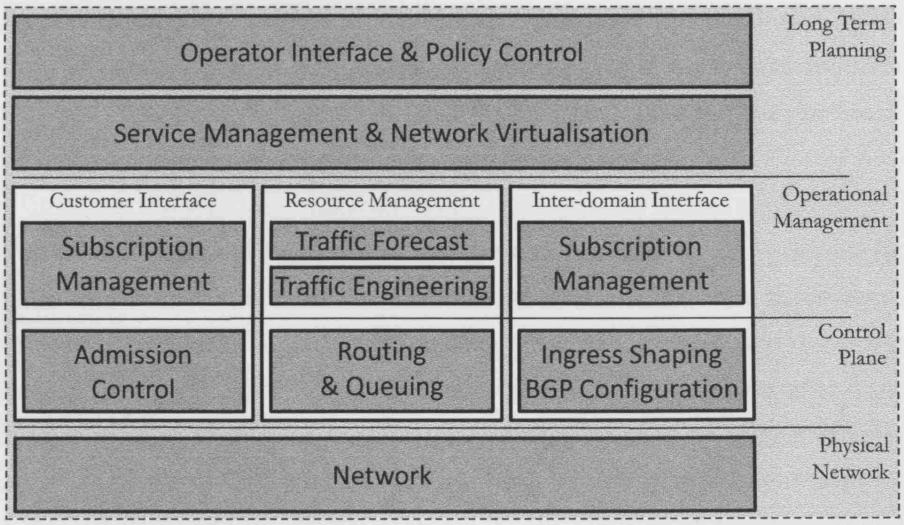


Figure 6.2: Framework for Subscription and Traffic Management based on [3]

The customer interface keeps track of the number of service units sold and either uses statistical multiplexing or hard partitioning to provide a forecast of resource usage.

Contained within resource management are traffic forecasting and traffic engineering. Traffic forecast is driven by both the customer subscription components and feedback from the traffic engineering component, providing monitoring feedback of actual resource usage. The traffic forecast component can make use of longer term traffic pattern analysis to forecast predictable events, which can then be taken into account by the operational components. The traffic engineering component houses functionality to configure the resources according to the requirements of traffic forecasting. Its detailed design depends on the underlying network structure.

The inter-domain interface contains components necessary for the management of inter-domain links and inter-provider agreements, it is similar to that used for customer management. Like for the customer interface, the subscription management component provides resource requirements to the network, but it can also act as a customer to other network providers and request inter-domain network capacity.

Service management component and network virtualisation is driven by high level service specifications and can extract data from the three operational components to derive requirements for each service and drive the network virtualisation. Virtualisation is the virtual separation of the (at least partially) unified physical network infrastructure by means of such technologies as MPLS and mt-OSPF. The service management component can make use of any network modelling abilities of the operational components. (i.e. those of the traffic engineering component that will be discussed later in more detail.)

Finally an oversight component is required that provides an interface to the network operator and via policy configuration or direct configuration of requirements.

As mentioned above, the diagram is fundamentally based on the TMN architecture. The main differences are in the definition of the service- and network management layers. In this diagram the service layer contains components similar to the "Business/Enterprise Management Layer" in TMN, in particular the management of service requirements and their translation to network layer goals. In contrast, the "Service Management Layer" of TMN contains the billing and service invocation that is housed in the "Subscription Management" in figure 6.2. The main argument for splitting functionality this way is that components that are responsible for billing and service subscription are directly interacting with network resource optimisation and traffic forecasting components in order to update resource usage and resource availability. The interaction between the tree components has to be bidirectional and is not client-server based, i.e. no one component drives the other two. It follows that all three components should be at in the same layer in the functional architecture. A second argument for placing service subscription into a lower layer is the need for it to drive the admission control components inside routers.

An in depth discussion of a complete management system with detailed decomposition of all necessary components, especially failure management and monitoring as well as interactions between the components is out of scope of this chapter. Several systems following this general design were proposed in the Information Society Technologies (IST) projects TEQUILA, MESCAL and AGAVE [GCG+03, FMB+03, BDO+06].
Tequila focuses on inter-domain management, Mescal extends Tequila with inter-domain capabilities and Agave attempts to integrate the previous platforms with ideas of multiple network planes and transparency with respect to individual engineering choices.

The remainder of this chapter will focus on the resource management component of the network management system. It will build a framework to identify how IPTE can be integrated with a management system and address some of the problems identified during this process.

6.5 Managing Traffic Engineering

The diagram in figure 6.3 shows a proposal for the internal structure of the traffic engineering component. Several sub-components are shown, *Network Configuration & Scheduling* as well as modules for specific traffic engineering mechanisms: *IPTE*, *MPLS-TE* and optical technologies such as *ASON*. The modules provide a layer of abstraction from the configuration details of the physical network. Each module contains algorithms

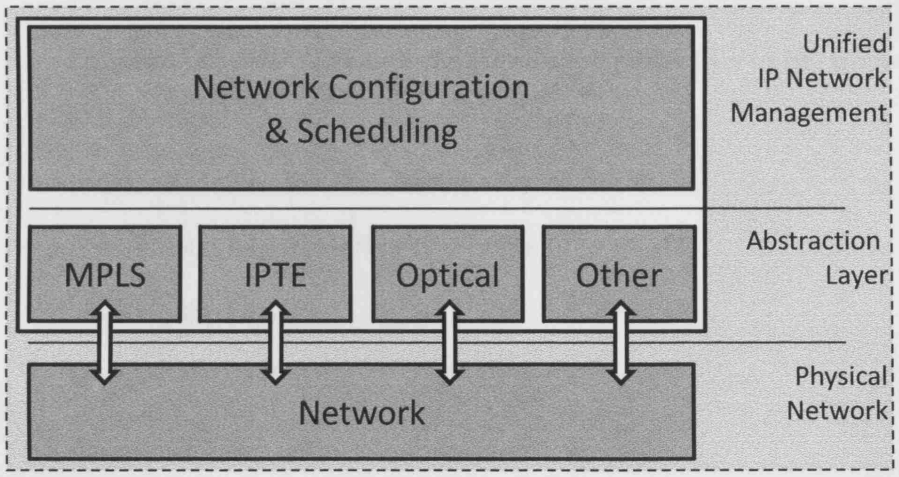


Figure 6.3: Decomposed Traffic Engineering

and interfaces allowing the translation of configuration requirements provided by network configuration and scheduling to the physical network infrastructure in place. This way, one management system can handle many layer 3 and below technologies using one system.

The IPTE module contains the link weight optimisation algorithm. Like the other modules, it is a passive block used by network configuration

to perform specific TE if required. For example, once called, the IPTE module is passed a traffic demand matrix as well as the current network topology by the Network Configuration and Scheduling (NCS) and computes a set of link weights. Computed weights are deposited in a link weight database inside the resource management block, until they are put into operation in the network by the scheduling component of network configuration. Similarly, the network configuration and scheduling can invoke any other TE modules that it has at its disposal in order to accomplish the required configurations.

6.5.1 Network Configuration and Scheduling

Objectives

- Manage traffic engineering requirements for the network
- Translate optimisation and configuration requirements to the traffic engineering modules
- Network configuration based on module output
- Prediction and monitoring of changes in
  - traffic demand
  - network topology
- Reconfiguration according to schedules or alerts

Basic Operation

The NCS component forms the control system of the resource optimisation block. It has two main purposes, handling computation requests to optimise network resources and scheduling the configuration of the network through use of the modules. The NCS uses the traffic forecast to predict traffic or infrastructure events and apply necessary configurations to the network.

The NCS acts as a mediator between the actual monitored and the predicted networks state as well as the QoS requirements of services. With all requirements in place, the modules give the NCS control over all available network resources through one unified interface. A network could then combine MPLS paths with IPTE optimised OSPF, where both techniques are used to their relative strengths. i.e. if an IPTE configuration fails to produce a result with the necessary fine control, an MPLS tunnel is deployed to carry the traffic that cannot be accommodated by IPTE. While this allows use of more than one technology in order to benefit from each technologies strength, it also avoids locking into one technology and allows the seamless migration from one technology to another without major changes to service planning, customer and inter-domain interfaces.

With time, the NCS can accumulate a large database of network configurations each suitable for one of many network states. For predictable patterns, each configuration can be applied pre-emptively to reconfigure the network in time to cause no disruptions. While daily repeating patterns are discovered more quickly, some more slowly evolving patterns may take longer to discover. This problem is mirrored in road traffic congestion control, where adaptive traffic light timings have been in place for some time. Yet the increasing complexity with higher traffic demands in cities has given rise to researching more adaptive systems, based on actual congestion measurements as well as past data [120]. Similarly, more sophisticated algorithms could accomplish the same task for IP network traffic and accumulate useful network configuration scenarios more quickly.

The stored configurations need to be updated when changes in topology occur, the size of the effect on any optimisation result is dependent on the impact the new link has on the network (i.e. a low capacity link is unlikely to have a large network-wide impact). It follows that the NCS has to adapt its already stored scenarios if possible and discard them if

they cannot be adapted. For example in the IPTE case, an "old" network configuration can be updated by passing it to the IPTE module together with the new topology. The old link weight settings can be applied as an initial condition and the network can be re-optimised. This should consume less time than re-computing from the beginning, although time taken and quality of result are dependent on the extent of the changes.

6.5.2 Optimisation Modules

Objectives

- Provide a layer of abstraction for network layer mechanism configuration, to enable the NCS to uniformly utilise all technologies for resource optimisation.
- Convert traffic demand matrix into corresponding configuration output, while honouring QoS constraints and optimising network capacity.
- Provide tools for the configuration of the modules network mechanisms.

Basic Operation

The diagram in figure 6.4 shows the interaction of the IPTE module with the NCS. It shows the relationship of one optimisation module with the NCS and relates to figure 6.3, which shows multiple modules.
Each optimisation module in figure 6.3 has to provide a configuration output that matches the given input requirements from the NCS. The module therefore contains an optimisation block as that described for IPTE in chapter 3. A corresponding module for MPLS would be less complex, since any route found by an optimisation algorithm could be implemented with a corresponding LSP's, making it easier to find an optimal solution. The MPLS optimisation module could also provide configurations for backup paths and other MPLS specific details.

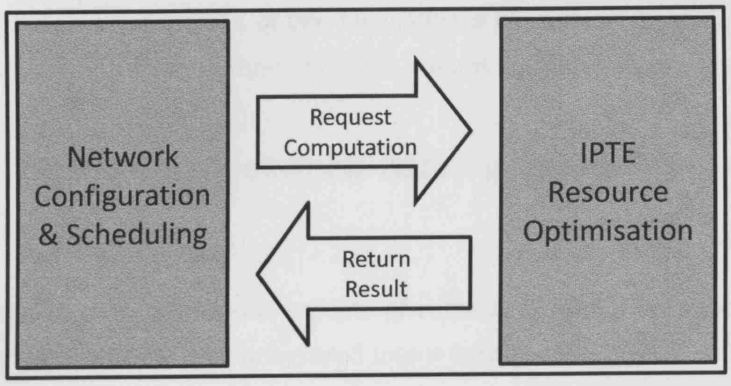


Figure 6.4: Module Interactions

Since modules will affect the amount of resources available in the network, each module has to pass the amount of resource usage to the NCS. The NCS can also configure the maximum resource spending for each optimisation request, especially in order to allow for specific configuration requests that do not encompass the whole traffic matrix (for example for an MPLS tunnel).

6.5.3 Integration of Modules

The ability to operate several modules in parallel potentially has many advantages. For example, MPLS can be used where explicit route pinning is the only viable solution, whereas IPTE can be used for the majority of the cases, hence combining the flexibility of MPLS route pinning with the less complex management and resilience of IPTE optimised OSPF. As long as a TE mechanism can integrate with the NCS block, it can be utilised by the management platform. In principle, even optical layer network planning technologies like ASON or GMPLS can be driven by such a system and be used to request more capacity and additional links between network nodes where required. However, since the details of such an integration have not been investigated, it is possible that such close integration is not possible without inter-module interaction. While the NCS can be used to make decisions to partition the available network resources and give individual jobs to the modules, it could also be conceived

to operate an optimisation of two mechanisms at once, such as simulated by Bagula [93] for MPLS and IGP link weight optimisation.

6.6 IPTE and Network Management

6.6.1 Outline

This section looks at various aspects of running IPTE in an operational setting, where it might be integrated into a management system. In order to react to dynamic changes in traffic patterns effectively, a link weight optimisation should be performed on the network, whenever traffic has changed sufficiently for optimisation to become necessary. A sufficient change can be identified by threshold values or by a trend indicating the a new traffic pattern.

It was demonstrated in the previous chapter, how the IPTE algorithm can be used to optimise the network for particular requirements. However, these requirements were static in nature. In order to effectively integrate with NCS and deal with dynamic changes in traffic, a more advanced management of IPTE results is required. The network management system is required to monitor the traffic on the network and the NCS has to match arising traffic patterns to the network configuration. If a pattern arises that is sufficiently different from the currently configured ones, a new link weight computation has to be triggered and implemented.

The pattern detection and matching is out of scope of this thesis, but much research has been done in this area, especially in order to detect malicious traffic and to block or control peer to peer traffic. Standard methods include port mapping as well as IP address matching against a list of known sources or sinks of traffic. More recently deep packet inspection techniques are also becoming viable. These were first covered in literature by Paxson [121] with recent advances, for example by Ma et al. [122], using the technique for traffic classification rather than its initial focus on intrusion detection.

Pattern detection works for long term changes where computation time is not a critical component. It also works for reoccurring short term patterns where the new pattern is already known. In this case, although the link weight computation algorithm may take too long to compute a suitable alternative configuration, solutions for these reoccurring scenarios can be pre-computed based on predicted or historic data and implemented immediately upon detection. For non-reoccurring scenarios the algorithm is limited to guessing: computations can be carried out by the management system pre-emptively. For example, alternative configurations could be triggered in case of a failure on each critical link in a network. Upon failure, the solution would be available immediately. Where traditionally, the network operators had to rely on the IGP to reconfigure itself, there is now a backup plan that restores not only the routing, but also some of the QoS requirements. Even for extreme traffic patterns, such as extraordinary events (new years, disasters, etc.) the management system can "learn" from past occurrences of similar patterns and a backup plan can be put in place that keeps at least critical services operational.

6.6.2 Modifying Link Weights in an Operational Network

There are problems associated with the modification of link weights in an operational network. Each single link weight update has to be flooded as a link state advertisement, causing a re-computation of the forwarding table in each router of the network. This is a disruptive process with the amount of disorder introduced into the network being a function of the number of modified link weights. It is therefore desirable to modify as few link weights as possible, as few times as possible. An example for a technique to limit the number of weight changes at each iteration is presented by Fortz et al. [117] as an extension to the link weight setting algorithm. However, any disruption caused by link weight modification is a deterrent from optimising too frequently and careful considerations have to precede an optimisation decision.

There are two different causes for link weight modifications to become necessary: scheduled management cycles to optimise and synchronise all network requirements as well as dynamic events that occur on shorter time-scales, within a cycle and require immediate attention.

Management cycles can cause extensive reconfiguration of the network according to long term changes in demand. However, it is also possible for a resource provisioning cycle to use little reconfiguration in the intra-domain network, if the demands can still be satisfied with little or no link weight modification. Although a more optimal solution may be found through an extensive reconfiguration, it should only be carried out if the additional cost caused by network disruption is factored into the cost trade-off. Network disruption can therefore be expressed as a threshold cost, causing the network to stay in a suboptimal state until reconfiguration is unavoidable. The value of the threshold is proportional to the size of the network (time to convergence after disruption) and number of link weight modifications that will be necessary to achieve re-optimisation (amount of disruption introduced). Finding solutions with as few weight changes as possible is therefore beneficial.

Smaller, dynamic changes in demand that occur within the Resource Provisioning Cycle (as outlined in the next section) have to cause minimal disruption to the network and are based on one or few link weight changes. This method is applied to optimise for new events with a concentrated demand pattern (release of a virus update, etc.) as well as link or node failures. Fluctuations in daily demand could also be addressed in this way, but it may be beneficial to take these into account by optimising weight settings for multiple demand matrices, because this technique does not require any periodic weight changes. Multiple demand matrix optimisation aims at selecting a weight setting solution that is suitable for a set of demand matrices that reoccur periodically.

6.6.3 Interaction with BGP

When deploying link weight optimisation in a network containing inter-domain links, there are several issues to be considered regarding interactions of IGP with BGP. All problems arise, because some of the decisions made by BGP can contradict the objectives of link weight optimisation.

The Internet routing table is not usually injected into the IGP. Doing this is considered bad practise, since the large number of injected routes can cause instability and slow convergence in the IGP. However, if inter-domain route information is not distributed in the local network, local nodes cannot route packets to inter-domain destinations. Two solutions exist to solve this problem. The first solution is most commonly deployed in small to medium size networks and involves configuring one or more "default gateways". A default gateway contains prefixes for whole or part of the inter-domain routing table and is injected into the IGP, which distributes the prefixes as a single entry linked to the gateway. A variant of this solution involves injecting partial inter-domain route information directly into the IGP. This can be useful if some large inter-domain sinks exist that require alternative treatment. The second solution is to ensure that all transit IGP routers are also BGP speakers. This would cause BGP to populate the necessary next hop information[2] in the router forwarding tables and the IGP is used to route to the next hop (Halabi [116]).

A problem arises when using link weight optimisation in conjunction with BGP. It involves conflicts with BGP route selection policies. If there is more than one connection to the Internet, the two BGP speakers use their routing policies to decide which part of the routing table each one will route depending on the inter-domain path metrics (LOCAL_PREF, AS_PATH, multi exit discriminator, etc.). Hence, if the the IGP config-

[2]In BGP terminology the next hop denotes the boarder router on the far side of an inter-domain link, i.e. it is an interface on first router in the next AS. This interface is logically part of the originating AS.

uration is based on default routes, traffic might be routed to the wrong boarder router and end up in a loop. A careless link weight optimisation, which is treating all boarder routers as potential gateways can thus introduce routing loops, if the BGP selection process is not overridden at the same time. The important point to consider is that if more than one egress exists towards the same inter-domain destination, BGP policies decide which egress is used and if the IGP routes traffic to the wrong egress a loop is created (Chapter 7 in [116]). This problem can be partially avoided by using the technique of deploying BGP on each transit router in the network discussed earlier, this would cause inter-domain traffic to be treated as local traffic with the next hop as sink. IGP routing would then be simply a matter of routing to the next hop.

A variant of the problem cannot be avoided by introducing BGP speakers on transit nodes and is recognised in the literature. It involves the BGP hot potato routing policy that selects the egress router based on IGP path length. Cerav-Erbas et al. [123] analyse the degradation caused to link weight optimisation if BGP modifies the next hop for a destination prefix after a link weight optimisation has caused the IGP path length to change. While their results show that degradation does indeed occur, it has already become clear in this discussion that IPTE has to go hand in hand with BGP policy in order to achieve good optimisation results.

Generally the caveats introduced by the interplay of BGP and link weight optimisation are of technical nature and can be avoided if a management platform is used to coordinate intra- and inter-domain resource optimisation. Taking the inter-domain path into consideration is not simply a matter of BGP, it is also part of a larger argument that was sidelined until now. Can there be an optimisation between costs of inter-domain links and costs of intra-domain links? Should network management prefer inter-domain load balancing over intra-domain load balancing or should intra-domain take preference? An attempt at solving this problem was made by Howarth et al. [3]. Ultimately, it might be instructive to re-

member that BGP is a policy based routing protocol, which provides an effective means for distributing the policy decisions of network operators. While it is capable of making routing decisions, it is also heavily configured to suit operators management decisions. Similarly, instead of providing an automated optimisation approach, the mechanisms of inter-, intra-domain path optimisation discussed here will enable network operators to make their own decisions.

6.6.4 Inter Plane Transitioning

When the network transitions to a new link weight state, the resulting transients in the converging routing tables will give rise to some out of order arrival and packet loss. This may be acceptable for certain types of traffic, especially low priority best-effort traffic. Where out of order-arrival and packet loss are not an option, the traffic can be transitioned smoothly by using a pre-configured, empty routing plane. This plane can be configured with the new link weight values and left to converge on its final routing table state, before traffic is transitioned from the old routing plane to the new one.

Transitioning is the process of changing the traffic ingress marking from the old- to the new routing plane and has to be done for all ingress routers that carry traffic on the routing plane. This ensures that no transient routing table convergence problems arise, however, it does not necessarily guarantee smooth transitioning. The process of switching from the old routing plane to the new one gives rise to a transient state in the network, where some traffic is on the new plane and some traffic is still carried on the old plane. This could potentially cause congestion on some links and therefore the order in which ingress routers are switched may be important.

6.7 Simulation of Inter Plane Transitioning

6.7.1 Theory

In order to determine how to best transition traffic from one plane to another, a model was built that allows the analysis of each intermediate step in the switch over. Using the cost function of the IPTE algorithm to assess the cost of traffic on each link, the cost of each intermediate step during the transition can be determined. A simulation can then show how this intermediate cost varies when a different switching order is chosen. This section shows a simple model and some preliminary results for the process of optimising the order of ingress router switching.

Figure 6.5 shows the cost of transitioning from routing plane 0 with cost ϕ_0 to routing plane 1 with cost ϕ_1. Cost ϕ_δ^r is an intermediate step in the process where x routers have already transitioned. The cost of transitioning ϕ_t can be modelled as follows. The total cost of transitioning is the sum of all ϕ_δ^r where the r identifies a set of all ingress router IDs arranged in a particular order. Any r identifies one ϕ_t (although not necessarily uniquely[3]) and is part of the set of solutions R, which is equal to all possible permutations of ordering the ingress routers, i.e. $R = N!$

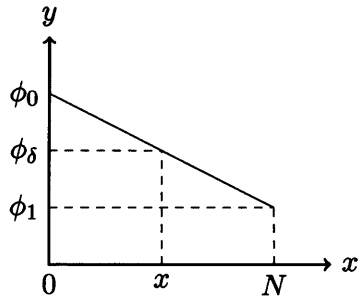


Figure 6.5: Cost of Plane Transition

[3]While each permutation of ingress node order r is unique, more than one r can lead to the same ϕ_t, since traffic from some routers may not affect traffic from other routers.

$$\phi_t^r = \sum_{x \in N} \phi_\delta^r(x) \tag{6.1}$$

It is also the area under the line in graph 6.5.

$$\phi_t^r = \sum_{x \in N} \phi_\delta^r(x) \approx \int_0^N \phi_\delta(x)dx \text{ for large } N \tag{6.2}$$

If the cost of transition were modelled as a straight line

$$\phi_\delta^r(x) = \phi_0 - \frac{\phi_0 - \phi_1}{N}x \tag{6.3}$$

equation (6.2) simplifies to the mean value of ϕ_δ multiplied by N.

$$\phi_t^r = N\frac{\phi_0 + \phi_1}{2} \tag{6.4}$$

6.7.2 Algorithm

The transitioning of traffic between routing planes can be optimised using a heuristic search, in a similar fashion to optimising link metrics. The objective is to minimise ϕ_t by rearranging the order of switching (i.e. the order of r).

$$\phi_t^r = \min_{r \in R} \sum_0^N \left(\phi_0 + \phi_1^r + \phi_2^r + ... + \phi_{N-1}^r + \phi_1\right) \tag{6.5}$$

1. Calculate ϕ_0 and ϕ_1
2. Initialise the transitioning by creating a random ordering of ingress nodes.
3. Calculate ϕ_t^r by summing the cost of each ϕ_δ transition from ϕ_0 to ϕ_1.
4. Randomly choose one pair of nodes who's order to swap for the next iteration.

5. Iterate steps 3 and 4 in order to find a solution with lower total cost, continue until no further improvement is seen or until number of iterations reaches a limit.

The algorithm deliberately does not look for any specific high or low value ϕ_δ, since there is no obvious correlation with the cost of a ϕ_δ and a particular switching order. In other words, it is not straight forward to identify a good candidate router to move to an earlier or later switching time without first gaining a deep understanding of the traffic distribution inside the network. Additionally, the problem space is $N!$, which can be very large.

6.7.3 Results

In order to test the theory and the algorithm for optimising the transition, first the transition between inverse capacity and unit weights has been simulated. In this case the cost of the new solution is larger than that of the old one, however, other factors than cost play into the decision whether or not to deploy a new set of weights. For the purpose of the simulation, this is not so relevant. The result for a 57% average loaded network transitioning from inverse capacity to unit weights is plotted in figure 6.6, showing ϕ_δ plotted against the number of transitioned routers. The transitioning algorithm was run for 2000 iterations. The first observation is that the intermediate steps have a lower ϕ_δ than ϕ_0 and ϕ_1, this is because only one plane contains traffic in this simulation and thus balancing traffic over two planes with different link metrics provides better load balancing. If the network contained a number of active routing planes, this effect is expected to be the inverse, i.e. higher cost should be incurred in the intermediate steps between ϕ_0 and ϕ_1. This simulation therefore just gives an indication that the approach is viable.

The result shows that the optimised transition provides lower intermediate costs than the random solution. It also shows a much smoother curve without intermediate valleys, which can be seen in the random solution

in several places. The valleys seem to indicate that the switching order of some ingress routers caused some higher intermediate load that disappeared again after further routers were switched. If the network were fully optimised with several routing planes, these are the events where transition congestion could occur.

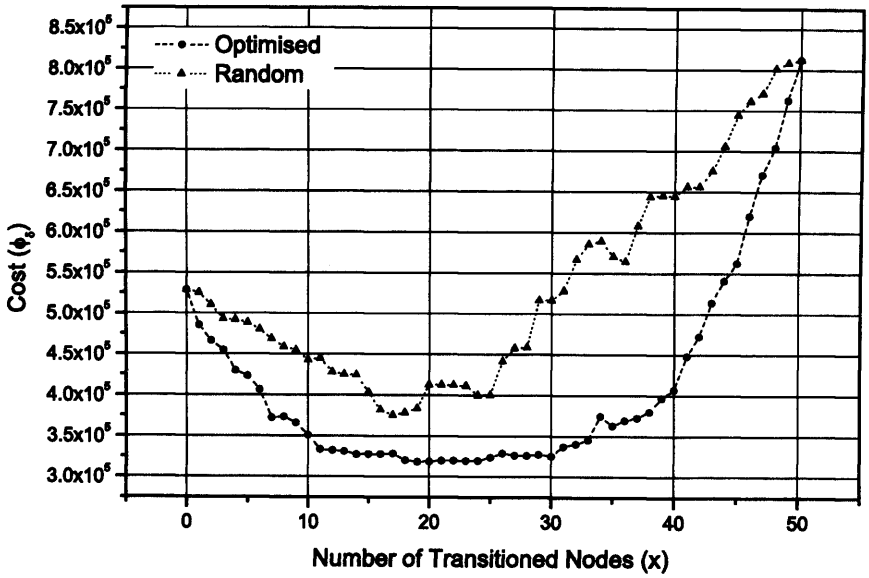


Figure 6.6: Load Transitioning from InverseCap to Unit Link Metrics

The second graph in figure 6.7 shows the cost of transitioning with increasing network plane load. The optimised transition is better than the random solution although the difference is not very large in comparison to the total value of ϕ_t. While the objective is to minimise ϕ_t, it is only possible to achieve this within limits, since each ϕ_δ always has to have a cost associated with it. The main reason for minimising ϕ_t is that it presents an effective means for achieving the desired reduction of complications during transitioning. Any spikes in the transition process are potential disruptions to network traffic and the more these spikes are reduced, the smoother the transition should be. The analysis and optimisation of the transitioning process described here, should provide the tools for effectively managing such a transition.

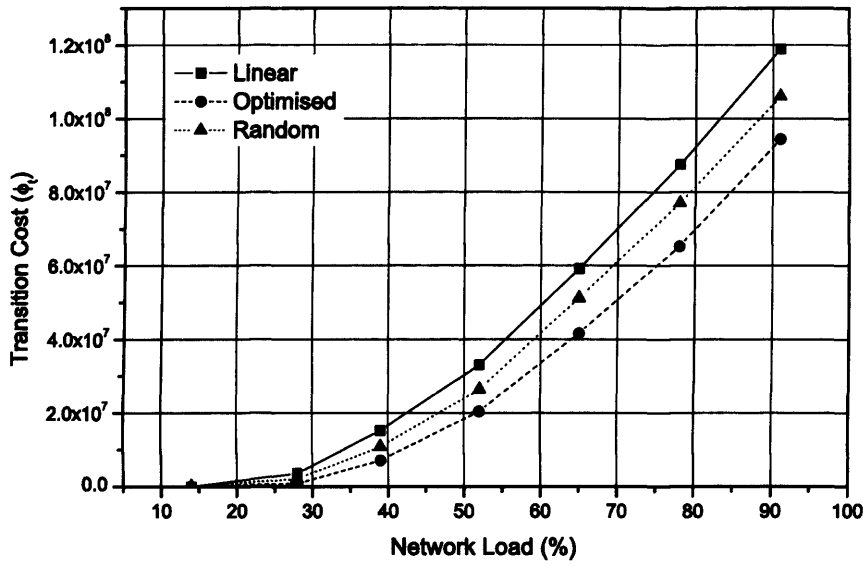


Figure 6.7: Load Transitioning with Changing Network Plane Load

6.8 Discussion

6.8.1 Summary

In order to establish the need for network management in multiservice IP networks, the fluctuating nature of traffic on such networks was pointed out. Not only does traffic fluctuate widely with time, multiservice traffic is also likely to vary in its QoS requirements.

Since IP networks lack inherent QoS differentiation, a management system has to coordinate the additional components required to implement QoS support for services. These components are admission control, traffic engineering and prioritised queuing. Additionally, a host of supporting functions are needed that maintain information about the network state and mediate between physical network and service requirements.

A simplified architecture for such a management system was presented in order to provide a framework for discussing the resource optimisation component that houses traffic engineering and traffic forecasting. It was

discussed how several traffic engineering mechanisms could coexist on one network infrastructure, all managed by the Network Configuration and Scheduling component inside resource optimisation. The coexistence would preserve the benefits of all deployed mechanisms as well as allowing seamless migration from one mechanism to another.

The use of IPTE in the context of the resource optimisation block was discussed in depth. The problem of network disruption through frequent link weight modifications was discussed from both intra- and inter-domain point of view. Dangers in careless deployment of link weight optimisation were pointed out. A solution to the problem of disruptions caused by link weight changes was presented in using several network planes to migrate traffic, rather than causing large disruptions to a "live" plane.

Finally it was recognised that migrating traffic between network planes could also cause disruptions, since transient states of the transition could cause link overloads. Hence, a strategy was developed for the smooth transitioning of traffic from one network plane to another. The strategy was simulated and preliminary results show that by carefully selecting the order of node switching, variations in intermediate network states can be reduced significantly.

6.8.2 Non-intrusive Management

The management platform described in this chapter outlines an integrated and flexible solution. While the IPTE algorithm provides a much improved means for traffic engineering the traditionally routed part of the network, it still leaves room for MPLS and other techniques.

The network management platform could become the pivotal point, enabling the quality guarantees necessary for multi-service networks as well as managing resilience and failures. Ideally, a management platform would not inhibit or remove the inherent features of IP networking, its distributed nature and its self managing capabilities. The IPTE platform

would merely guide the network routing algorithms and ensure a end-to-end oversight, so that objectives can be met that were previously only achievable by deploying MPLS. This way, the complexity of managing potentially thousands of MPLS tunnels is avoided.

The fact that *management platform* does not have to imply "centralised" is also an important consideration, which is often overlooked. While management platforms imply some sort of centralisation, they can be built in a resilient, decentralised way and because of their "offline" nature with respect to the time-scales of packet transport and routing decisions a 100% availablility of the managment components is not required. A well managed network should continue to operate within parameters throughout a period of management plane failure.

Chapter 7

Discussion and Conclusion

7.1 Summary

This thesis has investigated how intra-domain routing can be used to perform more complex traffic engineering tasks. The central cause of strain on today's IP network infrastructure stems from the industries striving to extend the network to carry voice and other advanced services that it was not designed for. The discussion should have made clear that although these services can be adopted at the application layer to work reasonably well (arguably well enough for many every-day applications), it is inadvisable to rely on current IP infrastructure for replacing circuit switched networks without having understood the implications this has on service reliability.

It was discussed that the key requirement for the future unified IP infrastructure is service differentiation in order to ensure the coexistence of several services with different QoS characteristics on the same network infrastructure. A definition of QoS was developed and the importance of QoS in IP networks, which divides opinions in the networking community was reiterated.

Several enabling components were identified, which are required in order to turn an IP network into a multiservice network with QoS support.

Techniques for engineering the network include network planning and traffic engineering, which are responsible for maximising the capabilities of the network and configuring it in such a way that traffic passing through the network experiences the least possible congestion. Admission control is added to limit the maximum amount of traffic that can enter the network, in order to avoid degradation of already running services. Finally, network management techniques were discussed, which act as the binding element for all components. Network management introduces an oversight where information on topology, traffic monitoring, forecasting and customer subscriptions can be kept together. The network management system forms the heart of a multiservice IP network that manages all engineering components and it is this oversight that can finally ensure that a level of QoS can be guaranteed for each service.

Special focus was placed on the traffic engineering element that forms the centre of this thesis. Existing QoS routing algorithms were identified, as well as algorithms for the optimisation of OSPF link weights. Modifying IP and IP routing protocols is difficult, because of the momentum created by the broad existing deployment of infrastructure. Tactics for avoiding the problem are found in mechanisms such as MPLS, but these techniques have their own problems especially in large scale deployment, because of scalability concerns and are therefore widely seen as intermediate solutions. It was pointed out that despite the existing algorithms for control plane QoS routing, the key to IP QoS support is more likely to be to be found in higher layers, management planes configuring a relatively "dumb" network backed up with admission control schemes to control the traffic amounts.

Two chapters were devoted to the IPTE traffic engineering algorithms that were developed by the author. The goal of the proposed traffic engineering approach is to compute a set of OSPF link weights to balance network load while honouring hop count, propagation delay, bandwidth or load balancing constraints of the traffic. The proposed solution is built on

classical OSPF routing and additionally uses the multi topology concepts that were recently proposed as an extension to OSPF. Using mt-OSPF multiple routing planes can be implemented. Each MT plane is assigned its own link metrics and can route traffic independently. Given a traffic demand matrix and the network topology, the algorithm computes a set of link weights using a search heuristic. The optimisation is cost function based, so that individual constraints can be taken account per routing plane. Being based on IP routing the TE solution is more lightweight than MPLS-TE in terms of state-information required to be maintained by the network and the associated management overhead for network configuration. Using this approach, QoS and other constraint information remains in the offline IPTE algorithms and thus no extra constraint awareness is required at layer 3. Since recent Cisco implementations of mt-OSPF and M-ISIS provide the required multi topology routing support, no major changes at the router level are required for the approach to be realised.

One chapter was devoted to the optimisation software that was built in order to carry out the validation of the algorithms. The chapter describes the software architecture and its inputs and outputs. I short description of the operation is also provided. Finally, some results from functional validation and optimisation of the algorithms efficiency are presented.

Extensive simulations were based on both BRITE generated Waxman topologies as well as Rocketfuel inferred topologies and used both flat and gravity models for traffic matrix generation. The results show that IPTE can successfully be used for distributing traffic more evenly across the network than either unit or inverse capacity link weights. Using up to four routing planes provides additional if small benefits over single plane IPTE. It was found that inverse capacity link weights may not be a good choice for the generated topologies, as even in topologies that should favour such weights, their performance was similar to unit link weights. The performance of IPTE on the two Rocketfuel inferred topologies, though

an improvement over unit and inverse capacity weights, was not as drastic as for the generated topologies. Better traffic matrix generation should improve these results, but in general the margin to inverse capacity should be smaller since these networks are built to suit inverse capacity link weight settings. The constraint based simulations show that delay and bandwidth constrained classes can be constructed effectively. The average end-to-end delay of the affected class was as low as that for unit link weights and much lower than the other non-delay constrained classes. Similarly, the bandwidth constraint class showed improvements over the mean utilisation over all other classes for both utilisation cases.

In order to establish the need for a flexible network management in multi-service IP networks, the fluctuating nature of traffic on such networks was pointed out. Not only does traffic fluctuate widely with time, multiservice traffic is also likely to vary in its QoS requirements.

Since IP networks lack inherent QoS differentiation, a management system has to coordinate the additional components required to implement QoS support for services. These components are admission control, traffic engineering and prioritised queuing. Additionally, a host of supporting functions are needed that maintain information about the network state and mediate between physical network and service requirements.

A simplified architecture for such a management system was presented in order to provide a framework for discussing the resource optimisation component that houses traffic engineering and traffic forecasting. It was discussed how several traffic engineering mechanisms could coexist on one network infrastructure, all managed by the Network Configuration and Scheduling component inside resource optimisation. The coexistence would preserve the benefits of all deployed mechanisms as well as allowing seamless migration from one mechanism to another.

The use of IPTE in the context of the resource optimisation block was discussed in depth. The problem of network disruption through frequent link weight modifications was discussed from both intra- and inter-domain point of view. Dangers in careless deployment of link weight optimisation were pointed out. A solution to the problem of disruptions caused by link weight changes was presented in using several network planes to migrate traffic, rather than causing large disruptions to a "live" plane.

Finally it was recognised that migrating traffic between network planes could also cause disruptions, since transient states of the transition could cause link overloads. Hence, a strategy was developed for the smooth transitioning of traffic from one network plane to another. The strategy was simulated with preliminary results showing that by carefully selecting the order of node switching, variations in intermediate network states can be reduced significantly.

7.2 Open Issues and Future Work

Any research conducted with sufficient depth raises more questions than it can answer. The deeper one's understanding of a problem, the more profound its many dimensions become and the less transparent possible solutions seem to be. This work is no different. What started as a simple idea quickly became an open ended thought process with ideas far out-reaching that what was covered through simulations or even more in depth analysis. Unfortunately, this means that many things are left undone. In the introduction, the objectives illuminated the general areas of interest, whereas the scope presented some coarse boundaries. This section outlines some of the concerns and opportunities that became apparent during design and implementation and after the work was completed.

The first category is based on work done for objective two, the proposal of a TE scheme based on a traditional IGP and concerns the optimisation algorithm itself. After the development of the algorithms, it became

apparent that there is room for algorithm improvements that would allow faster and better convergence especially for the optimisation with several routing planes. The optimisation for several routing planes could be started on the results for optimisations with less planes, in order to better make use of the already computed results. Care would have to be taken in avoiding that the algorithm is started in a local minimum. Generally, the more routing planes, the larger the likelihood that local minima are not escaped, because smaller amounts of traffic are moved with a single weight modification. Optimisation of convergence speed and arrival at good solutions is especially important if IPTE were to be deployed in a production network.

Tools to visualise the paths in the network and to summarise the information in order to make it comprehensible would greatly deepen the developers insights into the behaviour of the optimisation algorithms and thus allow him to target areas for improvements. These tools would have to be more sophisticated than annotated network diagrams, even a visual representation of network utilisation, such as link thickness, is likely to be insufficient, since it will quickly become unclear and confusing with increasing network size. More promising could be the development of a histogram based approach, categorising links in utilisation and capacity during the optimisation process.

The second category concerns objective three, the integration of IPTE with a network management platform. The interaction of the different TE mechanisms is largely unknown and only a few publications exist that combine MPLS and link weight optimisation (see section 2.5). Prediction and discovery of network events and the development of effective countermeasures as outlined in chapter 6 opens a large scope for further work in the monitoring area. In particular, it would be interesting to analyse the dynamic behaviour and the transients introduced by network failures and respective responses through reconfiguration. Ultimately, this

research could lead to an advanced self healing monitoring and management system as envisaged by projects like Tequila.

As an example where further work could lead, some inroads were made on the routing plane transitions. This idea shows much potential for further exploration. The simulations show two things: the idea of transitioning between routing planes is valid and allows the reconfiguration of link weights with reduced disruption and even the basic algorithm that was presented is sufficient in ensuring that the transition will be much smoother than a random transition. This means that if a more sophisticated means for managing link weight and network configurations were available, the approach could be shown to work for managing all of the predictable events, such as scheduled topology changes and traffic demand changes. As further evidence of potential community interest, Franois et al. [124] present an approach for gracefully updating link weights on a single routing plane, which received the 2007 INFOCOM best paper award. The authors use an approach similar to the optimisation of plane transitioning to avoid loops on transitioning a single plane to new link weights. The approach presented here is potentially even less disruptive than the approach presented by Franois et al..

The final category of outstanding work concerns deployment in a real network. Deployment of IPTE could take two forms: standalone and as part of the management framework from chapter 6. What is required for both solutions is an up to date traffic matrix, the network topology and a means of configuring link weights in routers. While there is no doubt that a standalone implementation would be easier to deploy, it would be more difficult to operate on a regular basis without the benefit of traffic forecasting mechanisms and without an automated router configuration mechanism, both of which would be supplied by a more complete management system. These difficulties would imply occasional re-optimisation of link weights in the standalone case and limit the potential of IPTE and especially the network plane transitioning, as a tool for adaptation

to short term traffic changes as well as network maintenance. A core network management system is therefore useful for implementation and would include some form of traffic forecasting and a means for automated router configuration.

7.3 Final Word

The work presented in this thesis has been driven by the increasingly complex applications that are being deployed over IP networks today, promoting ever more proposals for solutions that claim to do a better job than the currently implemented network. The motivation for this work stemmed from the hope of being able to preserve the distributed routing protocols of today's network and extend their strengths. As part of his contribution the author developed the IPTE algorithms and complemented them with an offline management framework in order to show how IPTE can be used to it's fullest potential. Some of the concepts and insights can be used to extend the current models of link weight optimisation, as well as multi topology IGP routing in the research community and maybe some of ideas will be a useful addition to operational routers and management platforms. Hopefully, this thesis has been able to show that IPTE is a valid, practical approach to solving some of the problems in intra-domain routing today.

Appendix A

The Authors Role in Research Projects

Several elements of this work have been part of the MESCAL and AGAVE projects. Essentially all the ideas in this thesis are the authors own, with exception of some initial thoughts on link weight modification stemming from the TEQUILA project and the functional architecture developed by the MESCAL project, parts of which were the foundation of chapter 6. However, the intra-domain traffic engineering architectural work in the chapter was not part of the project and was developed by the author alone. In addition to the work on the IPTE system, the author has attended in excess of 10 project meetings, where he has actively contributed towards the projects deliverables. These deliverables can be found in the "Technical Reports" section at the end of this thesis as well as in the "Publications" section. Aside from IPTE, sizeable contributions were made by the author on functional architectures, bi-directionality of traffic in two of the three inter-domain traffic engineering solutions of MESCAL and the design of the dynamic qBGP and its simulation for MESCAL that was published in [56]. Contributions were also made the offline inter-domain traffic engineering aspects of both MESCAL and AGAVE.

Publications

[ABE+05] H. Asgari, M. Boucadair, R. Egan, P. Morand, D. Griffin, J. Griem, P. Georgatsos, J. Spencer, G. Pavlou, and M.P. Howarth. Inter-Provider QoS Peering for IP Service Offering Across Multiple Domains. In *2nd Int. Workshop on Next Generation Networking Middleware (NGNM05), IFIP Networking Conference*, Waterloo, Canada, May 2005.

[Gri03] Jonas Griem. IP Traffic Engineering with Weighted Distributed Routing. In *London Communications Symposium (LCS)*, London, UK, September 2003.

[Gri04] Jonas Griem. Link Weight Optimisation with Quality of Service Support. In *London Communications Symposium (LCS)*, London, UK, September 2004.

[GSG+04] P. Georgatsos, J. Spencer, D. Griffin, P. Damilatis, H. Asgari, J. Griem, G. Pavlou, and P. Morand. Provider-level Service Agreements for Inter-domain QoS delivery. In *Fourth International Workshop on Advanced Internet Charging and QoS Technologies (ICQT04)*, September 2004.

[56] D. Griffin, J. Spencer, J. Griem, M. Boucadair, P. Morand, M. Howarth, N. Wang, G. Pavlou, A. Asgari, and P. Georgatsos. Interdomain routing through qos-class planes. *IEEE Communications Magazine*, Feb. 2007.

[111] M. Howarth, P. Flegkas, G. Pavlou, P. Trimintzios, D. Griffin, J. Griem, M. Boucadair, H. Asgari, and P. Georgatsos. An architectural framework for inter-domain quality of service provisioning. In *poster in Proc. IEEE/IFIP Int. Symp. on Integrated Network Management (IM)*, Nice, France, May 2005.

[3] M.P. Howarth, P. Flegkas, G. Pavlou, N. Wang, P. Trimintzios, D. Griffin, J. Griem, M. Boucadair, P. Morand, H. Asgari, and P. Georgatsos. Provisioning for inter-domain quality of service: the mescal approach. *IEEE Communications Magazine*, June 2005.

[WGS+07] N. Wang, D. Griffin, J. Spencer, J. Griem, J. Rodrguez Snchez, M. Boucadair, E. Mykoniati, B. Quoitin, M. Howarth amd G. Pavlou amd A. J. Elizondo, M. L. Garca Osma, and P. Georgatsos. A framework for lightweight qos provisioning: Network planes and parallel internets. In *IEEE/IFIP Symposium on Integrated Network Management (IM 2007)*, 2007.

Technical Reports

[AFM+04] H. Asgari, P. Flegkas, P. Morand, M. Boucadair, R. Egan, Mark Irons, D. Griffin, J. Griem, P. Trimintzios, P. Flegkas, P. Georgatsos, and T. Damilatis. D1.4: Issues in MESCAL Inter-Domain QoS Delivery: Technologies, Bi-directionality, Inter-operability, and Financial Settlements. Technical report, IST-MESCAL, 2004. Available from http://www.mescal.org.

[BDO+06] Mohamed Boucadair, Bruno Decraene, M. L. Garca Osma, A. J. Elizondo, J. Rodrguez Snchez, M. Boucadair, B. Decraene, B. Lemoine, E. Mykoniati, P. Georgatsos, D. Griffin, J. Spencer, J. Griem, N. Wang, M. Howarth, G. Pavlou, S. Georgoulas, and B. Quoitin. D1.1: Parallel Internets Framework. Technical report, IST-AGAVE, 2006. Available from http://www.ist-agave.org.

[BMA+05] M. Boucadair, P. Morand, H. Asgari, R. Egan, J. Griem, D. Griffin, J. Spencer, S. Georgoulas, K. H. Ho, M. Howarth, P. Trimintzios, N. Wang, P. Georgatsos, I. Liabotis, and E. Mykoniati. D3.2: Final Experimental Results: Validation and Performance Assessment of Algorithms and Protocols for Inter-domain QoS through Servicedriven Traffic Engineering. Technical report, IST-MESCAL, 2005. Available from http://www.mescal.org.

[FMB+03] P. Flegkas, P. Morand, M. Boucadair, P. Levis, Y. Noisette, N. Cantenot Egan, H. Asgari, D. Griffin, J. Griem, P. Trimintzios, N. Wang, M.P. Howarth, G. Pavlou, P. Georgatsos, T. Damilatis, G. Memenios, and D. Makris. D1.1: Specification of Business Models and a Functional Architecture for Inter-domain QoS Delivery. Technical report, IST-MESCAL, 2003. Available from http://www.mescal.org.

[GCG+03] Danny Goderis, Geoffry Crystallo, P. Georgatsos, T. Damilatis, M.Megalooikonomou, C. Jacquenet, M.Boucadair, S. Van Den Berghe, Pim Van Heuven, Eleni Mykoniati, D. Giannakopoulos, M. Maurogiorgis, H. Asgari, M. Irons, R. Egan, D. Griffin, M. Feng, J. Griem, P. Trimintzios, P.Flegkas, and G.Pavlou. D3.4: Final System Evaluation. Technical report, IST-TEQUILA, 2003. Available from http://www.ist-tequila.org.

[GSB+04] David Griffin, Jason Spencer, Mohamed Boucadair, Pierrick Morand, Hamid Asgari, Richard Egan, Jonas Griem, Paris Flegkas, Stelios Georgoulas, Kin (Roy) Hon Ho, Michael Howarth, George Pavlou, Panos Trimintzios, Ning Wang, Takis Damilatis, and Panos Georgatsos. D2.1: Implementation plan and high level engineering design of simulation models and testbed prototypes for inter-domain QoS delivery. Technical report, IST-MESCAL, 2004. Available from http://www.mescal.org.

[HMB+04] Michael Howarth, P. Morand, M. Boucadair, P. Levis, R. Egan, H. Asgari, D. Griffin, J. Griem, J. Spencer, P. Trimintzios, M. Howarth, N. Wang, P. Flegkas, K.-H. Ho, S. Georgoulas, G. Pavlou, P. Georgatsos, and T. Damilatis. MESCAL Deliverable D1.2: initial specification of protocols and algorithms for inter-domain sls management and traffic engineering for qos-based ip service delivery and their test requirements. Technical report, IST-MESCAL, 2004. Available from http://www.mescal.org.

[MBC+05] P. Morand, M. Boucadair, T. Coadic, P. Levis R., R. Egan, H. Asgari, D. Griffin, J. Griem, J. Spencer, M. Howarth, N. Wang, S. Georgoulas, K.H. Ho, P. Flegkas, P. Trimintzios, G. Pavlou, P. Georgatsos, E. Mykoniati, I. Liabotis, and T. Damilatis. D1.3: Final specification of protocols and algorithms for inter-domain SLS management and traffic engineering for QoS-based IP service delivery. Technical report, IST-MESCAL, 2005. Available from http://www.mescal.org.

[MOE+06] Eleni Mykoniati, M. L. Garca Osma, A. J. Elizondo, J. Rodrguez Snchez, M. Boucadair, B. Decraene, B. Lemoine, J.L. Le Roux, E. Mykoniati, P. Georgatsos, D. Griffin, J. Spencer, J. Griem, N. Wang, M. Amin, K. H. Ho, M. Howarth, G. Pavlou, B. Quoitin,

and O. Bonaventure. D2.1: Initial Specification of the Connectivity Service Provisioning Interface Components. Technical report, IST-AGAVE, 2006. Available from http://www.ist-agave.org.

[MPM+04] Eleni Mykoniati, P.Morand, M.Boucadair, H.Asgari, R.Egan, J.Griem, D.Griffin, J.Spencer, M.Howarth, R.Ho, S.Georgoulas, N.Wang, G.Pavlou, D.Makris, D.Manikis, K.Kavidopoulos, T.Damilatis, and M.Megaloikonomou. D3.1: Specification of Test Campaigns and Experimentation Plans for Performance Analysis and Prototype Validation. Technical report, IST-MESCAL, 2004. Available from http://www.mescal.org.

[WMO+06] Ning Wang, Eleni Mykoniati, M. L. Garca Osma, A. J. Elizondo, J. Rodrguez Snchez, M. Boucadair, B. Decraene, B. Lemoine, J.L. Le Roux, E. Mykoniati, P. Georgatsos, D. Griffin, J. Spencer, J. Griem, M. Amin, K. H. Ho, M. Howarth, G. Pavlou, B. Quoitin, and O. Bonaventure. D3.1: Initial Specification of Mechanisms, Algorithms and Protocols for Engineering the Parallel Internets. Technical report, IST-AGAVE, 2006. Available from http://www.ist-agave.org.

References

[1] Object Management Group (OMG). Recommendation M.3010, principles for a telecommunications management network (TMN), 1996.

[2] TINA-C. TINA v1.0 deliverables and specifications. URL `http://www.tinac.com/specifications/specifications.htm`.

[3] M.P. Howarth, P. Flegkas, G. Pavlou, N. Wang, P. Trimintzios, D. Griffin, J. Griem, M. Boucadair, P. Morand, H. Asgari, and P. Georgatsos. Provisioning for inter-domain quality of service: the mescal approach. *IEEE Communications Magazine*, June 2005.

[4] DE-CIX. DE-CIX Network Statistics, 2007. URL `http://www.de-cix.net/stats/`.

[5] Wenyu Jiang, K. Koguchi, and H. Schulzrinne. Qos evaluation of voip end-points. In *Communications, 2003. ICC '03. IEEE International Conference on*, volume 3, pages 1917–1921vol.3, 11-15 May 2003.

[6] Bernard Fortz and Mikkel Thorup. Internet traffic engineering by optimizing OSPF weights. In *INFOCOM (2)*, pages 519–528, 2000.

[7] Makarem Bamatraf and Mohamed Othman. Improved balancing heuristics for optimizing shortest path routing. *Computer Communications*, 30(7): 1513–1526, 2007. ISSN 0140-3664.

[8] Jun Wang, Yaling Yang, Li Xiao, and Klara Nahrstedt. Edge-based traffic engineering for ospf networks. *Computer Networks*, 48(4):605–625, 2005. ISSN 1389-1286.

[9] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault. Multi-topology (mt) routing in ospf, draft-ietf-ospf-mt-07.txt, November 2006. URL `http://www.ietf.org/internet-drafts/draft-ietf-ospf-mt-mib-00.txt`.

[10] British Telecom. BT 21st Century Network, 2006. URL `http://www.btplc.com/21CN/`.

[11] ftgroup. France Telecom NExT, 2005. URL `http://www.francetelecom.com/en/group/vision/strategy/next/`.

[12] Matt Beal. BT's 21st Century Network, Presentation at University College London, 2006. URL `http://www.btplc.com/21CN/WhatisBTsaying/Speechesandpresentations/Speeches.htm`.

[13] Robert Wood. *Next-Generation Network Services.* Number 1-58705-1591 in Networking Technology. Cisco Press, 2005.

[14] Robert M. Pirsig. *Zen and the Art of Motorcycle Maintenance: An Inquiry into Values.* Number 0-688-00230-7. Vintage, 1974. ISBN 0-688-00230-7.

[15] RealNetworks. Realvideo 10 technical overview (version 1), 2003. URL `http://docs.real.com/docs/rn/rv10/RV10TechOverview.pdf`.

[16] C. E. Perkins, O. Hodson, and V. J. Hardman. A survey of packet loss recovery techniques for streaming audio. *IEEE Network*, Volume: 12(5): Pages: 40–48, 1998.

[17] D. J. Thorne. Voip - the access dimension. *BT Technology Journal*, 19 No. 2, 2001.

[18] RealNetworks. Real player. URL `http://www.real.com`.

[19] Microsoft. Microsoft windows media player. URL `http://www.microsoft.com`.

[20] V. Cerf and R. Kahn. A protocol for packet network intercommunication. *IEEE Transactions on Communications*, Volume: COM-22 No. 5:Pages: 637–648, 1974.

[21] Henning Schulzrinne, S. Casner, R. Frederick, and V. Jacobsen. RFC1889: rtp: A transport protocol for real-time applications. standards track, 1996.

[22] Iso Iec. Open distributed processing- reference model - part 2: Foundations international standard 10746-2 itu-t recommendation x.902, 1995.

[23] Andreas Vogel, Brigitte Kerhervé, Gregor v. Bochmann, and Jan Gecsei. Distributed multimedia applications and quality of service: a survey. In *CASCON '94: Proceedings of the 1994 conference of the Centre for Advanced Studies on Collaborative research*, page 71. IBM Press, 1994.

[24] A. Bouch and M. A. Sasse. Network quality of service: What do users need? In *4th International distributed conference (IDC) Pages: 78-90*, 1999.

[25] Michael Hauben. History of the ARPANET. URL `http://www.dei.isep.ipp.pt/docs/arpa.html`.

[26] Chuck Fraleigh, Fouad Tobagi, and Christophe Diot. Provisioning IP Backbone Networks to Support Latency Sensitive Traffic. In *IEEE INFOCOM*, April 2003.

[27] Baek-Young Choi, Sue Moon, Zhi-Li. Zhang, Konstantina Papagiannaki, and Christophe Diot. Analysis of point-to-point packet delay in an operational network. In *Proceedings of IEEE INFOCOM, Hong Kong*, March, 2004.

[28] Michael Menth, Ruediger Martin, and Joachim Charzinski. Capacity Overprovisioning for Networks with Resilience Requirements. In *Proceedings of SIGCOMM*, Pisa, Italy, September 2006.

[29] David J. Houck and Gopal Meempat. Centralized call admission control and load balancing for voice over i. In *SPIE*, volume 4211 of *Internet Quality and Performance and Control of Network Systems*, pages 105–112, February 2001.

[30] Michael Welzl. *Network Congestion Control.* Wiley, 2005. ISBN 0-470-02528-X. ISBN: 0-470-02528-X.

[31] Soren Asmussen. *Applied Probability and Queues.* Wiley, 1987. ISBN 0-471-91173-9. ISBN: 0-471-91173-9.

[32] Christian Huitema. *Routing in the Internet.* Prentice Hall, 2nd edition, 1995. ISBN 0-13-022647-5. ISBN: 0-13-022647-5.

[33] J. Moy. RFC 2328: Open shortest path first version 2. standards track., 1998.

[34] D. Oran. RFC 1142: OSI IS-IS intra-domain routing. informational protocol, feb 1990.

[35] R. W. Callon. RFC1195: Use of OSI IS-IS for routing in TCP/IP and dual environments, 1990.

[36] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik Volume: 1 Pages: 269-271*, 1959.

[37] J.P.G. Sterbenz. Intelligence in future broadband networks: challenges and opportunities in high-speed active networking. *IEEE Broadband Communications*, pages 2–1–2–7, 2002.

[38] Lily Cheng, Yang Cao, John Ellson, Anwar Elwalid, Takeshi Hashimoto, Hirokazu Ishimatsu, Admela Jukan, Li Mo, Ananth Nagarajan, Yoshihito Oyama, Lijun Qian, Iraj Saniee, Ed Snyder, Shinya Tanaka, Maarten Vissers, Indra Widjaja, Yangguang Xu, and Susumu Yoneda. A framework for internet network engineering, 2001. URL `http://ftp.ist.utl.pt/pub/pub/drafts/draft-cheng-network-engineering-framework-01.txt`.

[39] A. Jajszczyk. Automatically switched optical networks: benefits and requirements. *IEEE Communications Magazine*, Volume: 43(2):Pages: 10–15, 2005.

[40] Lou Berger, Peter Ashwood-Smith, Ayan Banerjee, Greg Bernstein, John Drake, Yanhe Fan, Kireeti Kompella, Jonathan P. Lang, Eric Mannie, Bala Rajagopalan, Yakov Rekhter, Debanjan Saha, Vishal Sharma, George Swallow, and Z. Bo Tang. RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description, 2003.

[41] Nic Larkin. ASON and GMPLS - the battle for the optical control plane, 2002. URL `http://www.dataconnection.com/products/whitepapers.htm`.

[42] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. RFC 2475: An architecture for differentiated services. proposed standard services, 1998.

[43] R. Braden, D. Clark, and S. Shenker. RFC1633: Integrated Services in the Internet Architecture: an Overview. Informational, 1994.

[44] D. Goderis, S. Van den Bosch, and Y. T'Joens. A service-centric ip quality of service architecture for next generation networks. In *IEEE/IFIP Network Operations and Management Symposium (NOMS) Volume: 8, no. 1 Pages: 139-154*, 2002.

[45] Cisco Systems. Diffserv - the scalable end-to-end QoS model, 2001. URL `http://www.cisco.com/warp/public/cc/pd/iosw/prodlit/difsewp.pdf`.

[46] F. Baker, C. Iturralde, F. Le Faucheur, and B. Davie. RFC3175: Aggregation of RSVP for IPv4 and IPv6 Reservations. Standards Track, 2001.

[47] Cisco IOS Manual. Load splitting ip multicast traffic over ecmp. URL `http://www.cisco.com/en/US/products/ps6441/products_configuration_guide_chapter09186a008082ac4b.html`.

[48] A. Iselt, A. Kirstadter, A. Pardigon, and T .Schwabe. Resilient routing using MPLS and ECMP. In *Workshop on High Performance Switching and Routing*, pages 345– 349, 2004.

[49] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. RFC3272: Overview and principles of internet traffic engineering, 2002.

[50] Srikanth Kandula, Dina Katabi, Shantanu Sinha, and Arthur Berger. Dynamic load balancing without packet reordering. *SIGCOMM Computer Communications Review*, 37(2):51–62, 2007. ISSN 0146-4833.

[51] M. Allman, V. Paxson, and W. Stevens. RFC 2581: TCP Congestion Control. Standards Track, 1999.

[52] D. Bertsekas. Dynamic behavior of shortest path routing algorithms for communication networks. *Automatic Control, IEEE Transactions*, Volume: 27(1):Pages: 60–74, 1982.

[53] Zheng Wang and Jon Crowcroft. Analysis of shortest-path routing algorithms in a dynamic network environment. *Computer Communication Review*, Volume: 22(Number: 2):Pages: 63–71, 1992.

[54] A. Khanna and J. Zinky. The revised ARPANET routing metric. In *ACM SIGCOMM Pages: 45-56*, 1989.

[55] J. Moy. RFC 1583: Open shortest path first version 2. standards track., 1994.

[56] D. Griffin, J. Spencer, J. Griem, M. Boucadair, P. Morand, M. Howarth, N. Wang, G. Pavlou, A. Asgari, and P. Georgatsos. Interdomain routing through qos-class planes. *IEEE Communications Magazine*, Feb. 2007.

[57] Tony Przygienda, Naiming Shen, and Nischal Sheth. M-ISIS: Multi topology (mt) routing in is-is, draft-ietf-isis-wg-multi-topology-11.txt, October 2005. URL `http://www.ietf.org/internet-drafts/draft-ietf-isis-wg-multi-topology-11.txt`.

[58] Cisco Systems. IS-IS multi-topology, 2003. URL `http://www.cisco.com/warp/public/cc/pd/iosw/prodlit/isismwp.pdf`.

[59] Cisco Systems. Multi-topology routing (MTR), 2007. URL `http://www.cisco.com/en/US/products/ps6922/products_feature_guide09186a00807c64b8.html`.

[60] E. Rosen, A. Viswanathan, and R. Callon. RFC 3031: multiprotocol label switching architecture, standards track, 2001.

[61] John Kenney and Vijay Samalam. Traffic Management Specification Version 4.1. Technical report, ATM Forum, 1999.

[62] F. Le Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, and J. Heinanen. Multi-Protocol Label Switching (MPLS) Support of Differentiated Services. RFC 3270 (Proposed Standard), May 2002.

[63] W. Ben-Ameur, N. Michel, B. Liau, J. Geffard, and E. Gourdin. Routing strategies for IP-Networks. *Telektronikk Magazine*, Volume: 2/3, 2001.

[64] A. Farrel, D. Papadimitriou, J.-P. Vasseur, and A. Ayyangar. Encoding of Attributes for Multiprotocol Label Switching (MPLS) Label Switched Path (LSP) Establishment Using Resource ReserVation Protocol-Traffic Engineering (RSVP-TE). RFC 4420 (Proposed Standard), February 2006.

[65] P. Pan, G. Swallow, and A. Atlas. Fast Reroute Extensions to RSVP-TE for LSP Tunnels. RFC 4090 (Proposed Standard), May 2005.

[66] A. Farrel. Fault Tolerance for the Label Distribution Protocol (LDP). RFC 3479 (Proposed Standard), February 2003.

[67] George Apostolopoulos, Roch Guerin, Sanjay Kamat, and Satis K. Tripathi. Quality of service based routing: A performance perspective. In *Proceedings of SIGCOMM*, pages 17–28, 1998.

[68] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. RFC2386: A framework for qos-based routing in the internet, 1998.

[69] R. Guerin, A. Orda, and D. Williams. RFC2676: QoS routing mechanisms and OSPF extensions, 1996.

[70] Zhang, Sanchez, Salkewicz, and Crawley. Quality of service path first routing, 1996. URL `http://tools.ietf.org/html/draft-zhang-qos-ospf-00`. draft-zhang-qos-ospf-00.txt.

[71] K. Kar, M. Kodialam, and T. Lakshman. Minimum interference routing of bandwidth guaranteed tunnels with mpls traffic engineering applications. *IEEE Journal on Selected Areas in Communications*, 18:12, 2000.

[72] C.L.P. Bin Wang; Xu Su; Chen. A new bandwidth guaranteed routing algorithm for mpls traffic engineering. *ICC*, 2:1001– 1005, 2002.

[73] E. Atteo, S. Avallone, and S. P. Romano. A link weight assignment algorithm for traffic-engineered networks. *Computer Networks*, 50(13):2286–2294, 2006. ISSN 1389-1286.

[74] P. Van Mieghem and F. A. Kuipers. On the complexity of QoS routing. *Computer Communications*, 26(4):376–387, 2003.

[75] H. de Neve and P. van Mieghem. A multiple quality of service routing algorithm for pnni. In *ATM Workshop Proceedings, 1998 IEEE*, pages 324–328, 26-29 May 1998.

[76] Yong Cui, Jianping Wu, and Ke Xu. Precomputation for intra-domain QoS routing. *Computer Network ISDN Systems*, 47(6):923–937, 2005. ISSN 0169-7552.

[77] Turgay Korkmaz and Marwan Krunz. A randomized algorithm for finding a path subject to multiple QoS requirements. *Computer Networks (Amsterdam, Netherlands: 1999)*, 36(2–3):251–268, 2001.

[78] Yanxing Zheng, Turgay Korkmaz, Wenhua Dou, and Jing Tian. Highly responsive and efficient QoS routing using pre- and on-demand computations along with a new normal measure. *Computer Networks*, 50(18): 3743–3762, 2006. ISSN 1389-1286.

[79] S. Chen and K. Nahrstedt. An overview of quality-of-service routing for the next generation high-speed networks: Problems and solutions. *IEEE Network Magazine Special Issue on Transmission and Distribution*, 1998.

[80] Yanxing Zheng, Wenhua Dou, Jing Tian, and Mingyan Xiao. An overview of research on QoS routing. *Advanced Parallel Processing Technologies*, pages 387–397, 2003.

[81] X. Xiao and L. M. Ni. Internet QoS: A big picture. *IEEE Network*, 13(2): 8–18, Mar 1999.

[82] Cisco Systems. IP2R: OSPF commands, 2004. URL `http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fiprrpr/1rfospf.htm`.

[83] M. Ericsson, M. G. C. Resende, and P. M. Pardalos. A genetic algorithm for the weight setting problem in OSPF routing. Technical report, AT&T Shannon Laboratory 180 Park Avenue Florham Park, NJ 07932, 2001.

[84] Luciana S. Buriol, Mauricio G. C. Resende, Celso C. Ribeiro, and Mikkel. A memetic algorithm for ospf routing. In *INFORMS (6)*, pages 187–188, 2002.

[85] Tao Ye, Hema Tahilramani Kaur, Shivkumar Kalyanaraman, Kenneth S. Vastola, and Saroj Yadav. Optimization of OSPF Weights Using On-line Simulation. In *Tenth Annual International Workshop on Quality of Service (IWQoS)*, 2002.

[86] A. Riedl. A hybrid genetic algorithm for routing optimization in ip networks utilizing bandwidth and delay metrics. In *IEEE Workshop on IP Operations and Management (IPOM)*, Dallas, USA, 2002.

[87] Cisco Systems. Enhanced interior gateway protocol. URL `http://www.cisco.com/univercd/cc/td/doc/cisintwk/itodoc/enigrp.htm`.

[88] Bernard Fortz and Mikkel Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, pages 20(4):756–767, 2002.

[89] Bernhard Fortz and Mikkel Thorup. Robust optimization of OSPF/IS-IS weights. In *Proceedings of INOC*, pages 225–230, 2003.

[90] B. Quoitin and S. Uhlig. Modeling the routing of an autonomous system with c-bgp. *Network, IEEE*, 19(6):12–19, Nov.-Dec. 2005.

[91] R. Teixeira and J. Rexford. Managing routing disruptions in internet service provider networks. *Communications Magazine, IEEE*, 44(3):160–165, March 2006.

[92] Pedro Sousa, Miguel Rocha, Miguel Rio, and Paulo Cortez. Efficient OSPF Weight Allocation for Intra-domain QoS Optimization. In Gerard Parr, David Malone, and Michaeal O Foghlu, editors, 6_{th} *IEEE International Workshop on IP Operations and Management, IPOM 2006*, pages 37–48, Dublin, Ireland, October 2006. LNCS 4268, Springer-Verlag.

[93] Antoine B. Bagula. Online traffic engineering: A hybrid igp+mpls routing approach. *Quality of Service in the Emerging Networking Panorama*, pages 134–143, 2001.

[94] M. Piro, A. Szentesi, J. Harmatos, A. Juettner, P. Gajowniczek, and S. Kozdrowski. On open shortest path first related network optimisation problems. *Performance Evaluation*, 48(1-4):201–223, 2002. ISSN 0166-5316.

[95] M. P. Pettersson, R. Szymanek, and K. Kuchcinski. A CP-LP hybrid method for unique shortest path routing optimization. In *International Network Optimization Conference*, Spa, Belgium, 2007.

[96] Shekhar Srivastava, Gaurav Agrawal, Deep Medhi, and Michal Pioro. Determining feasible link weight systems under various objectives for ospf networks. *IEEE eTransactions on Network and Service Management*, 2005.

[97] G. Retvari and T. Cinkler. Practical ospf traffic engineering. *Communications Letters, IEEE*, 8(11):689–691, Nov. 2004.

[98] Dean H. Lorenz, Ariel Orda, Danny Raz, , and Yuval Shavitt. How good can IP routing be? Technical Report 2001-17, 2001.

[99] M. Roughan, M. Thorup, and Y. Zhang. Traffic engineering with estimated traffic matrices. In *USENIX/ACM Internet Measurement Conference (Miami, Florida)*, pages 248–258, October 2003.

[100] Anja Feldmann, Albert G. Greenberg, Carsten Lund, Nick Reingold, Jennifer Rexford, and Fred True. Deriving traffic demands for operational IP networks: methodology and experience. In *Proceedings of SIGCOMM*, pages 257–270, 2000.

[101] A. Medina, N. Taft, K.Salamatian, S. Bhattacharyyam, and C. Diot. Traffic matrix estimation techniques: Existing techniques compared and new directions. In *Proceedings of SIGCOMM*, Pittsburgh, PA, August 2002.

[102] Jacek P. Kowalski and Bob Warfield. Modelling traffic demand between nodes in a telecommunications network. In *Australian Telecommunications and Networks Conference*, Sydney, Australia, December 1995.

[103] Y. Vardy. Network tomography: Estimating source-destination traffic intensities from link data. *Journal of the American Statistical Association*, 91:365–377, 1996.

[104] Nick Duffield. Simple network performance tomography. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 210–215, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-773-7.

[105] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale ip traffic matrices from link loads. In *ACM SIGMETRICS*, San Diego, CA, June 2003.

[106] V.P. Kumar, T.V. Lakshman, and D. Stiliadis. Beyond best effort: router architectures for the differentiated services of tomorrow's internet. *Communications Magazine, IEEE*, 36(5):152–164, May 1998.

[107] Lundy Lewis. *Service Level Management for Enterprise Networks.* Artech House, 1999. ISBN 1-58053-016-8. ISBN: 1-58053-016-8.

[108] Object Management Group (OMG). Common object request broker architecture (CORBA/IIOP), , revision 3.0.3, December 2006.

[109] Cisco. Cisco CRS-1 series carrier routing system XML API guide. URL `http://www.cisco.com/en/US/docs/ios_xr_sw/iosxr_r2.0/xml/program/guide/FCSxml.pdf`.

[110] P. Trimintzios, I. Andrikopoulos, G. Pavlou, P. Flegkas, D. Griffin, P. Georgatsos, D. Goderis, Y. TJoens, L. Georgiadis, C. Jacquenet, and

R. Egan. A management and control architecture for providing IP differentiated services in MPLS-based networks. *IEEE Communications Magazine*, 39(5):80–88, 2001.

[111] M. Howarth, P. Flegkas, G. Pavlou, P. Trimintzios, D. Griffin, J. Griem, M. Boucadair, H. Asgari, and P. Georgatsos. An architectural framework for inter-domain quality of service provisioning. In *poster in Proc. IEEE/IFIP Int. Symp. on Integrated Network Management (IM)*, Nice, France, May 2005.

[112] A. Jttner, A. Szentesi, J. Harmatos, M. Piro, and P. Gajowniczek. On Solvability of an OSPF Routing Problem. In *15th Nordoc Teletraffic Seminar*, 2000.

[113] Q. Ma and P. Steenkiste. Quality-of-service routing for traffic with performance guarantees. In *IFIP Fifth International Workshopon Quality of Service, NY, NY*, pages 115–126, 1997.

[114] P. Trimintzios, L. Georgiadis, G. Pavlou, D. Griffin, D. Cavalcanti, P. Georgatsos, and C. Jacquenet. Engineering the Multi-Service Internet: MPLS and IP-based Techniques. In *IEEE International Conference on Telecommunications (ICT) Volume: 3 Pages: 129-134*, 2001.

[115] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. RFC2702: Requirements for Traffic Engineering over MPLS. Informational, 1999.

[116] Bassam Halabi. *Internet Routing Architectures.* Cisco Press, 1996. ISBN 1-56205-652-2. ISBN: 1-56205-652-2.

[117] Bernard Fortz, Jennifer Rexford, and Mikkel Thorup. Traffic engineering with traditional ip routing protocols. *IEEE Communications Magazine*, pages 118–124, October 2002.

[118] A. Medina, A. Lakhina, I. Matta, and J. W. Byers. BRITE: An Approach to Universal Topology Generation. In *EEE MASCOTS*, 2001.

[119] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. Inferring link weights using end-to-end measurements. In *ACM SIGCOMM Internet Measurement Workshop*, 2002.

[120] John Fawcett and Peter Robinson. Adaptive routing for road traffic. *IEEE Computer Graphaphic Applications*, 20(3):46–53, 2000. ISSN 0272-1716.

[121] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435–2463, 1999. ISSN 1389-1286.

[122] Justin Ma, Kirill Levchenko, Christian Kreibich, Stefan Savage, and Geoffrey M. Voelker. Unexpected means of protocol inference. In *IMC '06: Proceedings of the 6th ACM SIGCOMM on Internet measurement*, pages 313–326, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-561-4.

[123] Selin Cerav-Erbas, Olivier Delcourt, Bernard Fortz, and Bruno Quoitin. The interaction of IGP weight optimization with BGP. *ICISP*, 0:9, 2006.

[124] Pierre Franois, Mike Shand, and Olivier Bonaventure. Disruption-free topology reconfiguration in OSPF networks. In *IEEE INFOCOM*, Anchorage, USA, May 2007. INFOCOM 2007 Best Paper Award.